

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 项目名称： | 仓库管理系统 |
| 姓 名： | 李泽贤 |
| 专 业： | CS |
| 班 级： | 1703 |
| 学 号： | U201714600 |
| 指导教师： | 丁晓锋 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2020年6月18日

**教师评分页**

|  |  |
| --- | --- |
| 子目标 | 子目标评分 |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
|  |  |

目 录

[1 课程任务概述 1](#_Toc4559)

[2 软件的基本功能学习 1](#_Toc28652)

[2.1 任务要求 1](#_Toc21490)

[2.2 完成过程 1](#_Toc17406)

[2.3任务总结 3](#_Toc6646)

[3交互式使用SQL语言 3](#_Toc30313)

[3.1 任务要求 3](#_Toc5104)

[3.2 完成过程 3](#_Toc30954)

[3.3任务总结 16](#_Toc20155)

[4 仓储管理系统 16](#_Toc18110)

[4.1系统设计目标 16](#_Toc6978)

[4.2 需求分析 16](#_Toc11576)

[4.3 总体设计 17](#_Toc14659)

[4.4 数据库设计 17](#_Toc15887)

[4.5 详细设计与实现 18](#_Toc10452)

[4.6 系统测试 21](#_Toc24884)

[4.7 系统设计与实现总结 35](#_Toc24954)

[5 课程总结 35](#_Toc27175)

[6 附录 36](#_Toc25235)

[疫情火车sql 36](#_Toc29990)

[存储管理sql 59](#_Toc28760)

# 1 课程任务概述

通过上机实践，熟悉一种大型数据库管理系统，了解DBMS的体系结构。

熟练掌握SQL的数据定义、数据操纵和数据控制语言的运用。

熟悉数据库应用系统的设计方法和开发过程。

# 软件的基本功能学习

## 2.1 任务要求

软件的基本功能学习。

1）练习SQL Server或其他某个主流关系数据库管理系统软件的备份方式，要求要有通过数据库的软件功能进行的备份和通过文件形式的脱机备份。

2）练习在新增的数据库上增加用户并配置权限的操作。

## 2.2 完成过程

### 2.2.1 关系数据库管理系统软件备份

1)软件逻辑备份

备份sql语句，一键还原备份。如图2.1

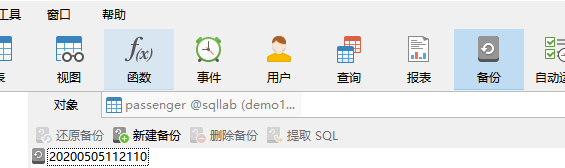


图2.1 软件备份

2)脱机备份

一、MySQL数据库文件介绍

MySQL数据库文件包括MySQL（server）所建数据库文件和MySQL（server）所用存储引擎创建的数据库文件。

1、MySQL（server）创建并管理的数据库文件：

.frm文件：存储数据表的框架结构，文件名与表名相同，每个表对应一个同名frm文件，与操作系统和存储引擎无关，即不管MySQL运行在何种操作系统上，使用何种存储引擎，都有这个文件。

除了必有的.frm文件，根据MySQL所使用的存储引擎的不同（MySQL常用的两个存储引擎是MyISAM和InnoDB），存储引擎会创建各自不同的数据库文件。

2、MyISAM数据库表文件：

.MYD文件：即MY Data，表数据文件

.MYI文件：即MY Index，索引文件

.log文件：日志文件

3、InnoDB采用表空间（tablespace）来管理数据，存储表数据和索引，

InnoDB数据库文件（即InnoDB文件集，ib-file set）：

ibdata1、ibdata2等：系统表共享空间文件，存储InnoDB系统信息和用户数据库表数据和索引，所有表共用。

.ibd文件：单表表空间文件，每个表使用一个表空间文件（file per table），存放用户数据库表数据和索引。

日志文件： ib\_logfile1、ib\_logfile2当Innodb表损坏时，可以依靠这些日志文件来恢复。

二、我的mysql数据存放在 Mysql Server x.x\data\下。对于本次实验，对数据库文件夹，ibdata1， ib\_logfile0， ib\_logfile1b1，ib\_buffer\_pool，mysql.ibd备份。恢复时进行覆盖即可。

### 2.2.2数据库上增加用户并配置权限

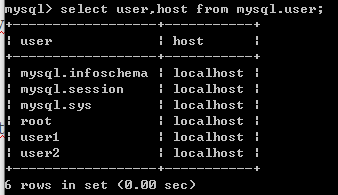
建立新用户

mysql -u root -p

create user 'user1'@'localhost' identified by '5953';

create user 'user2'@'localhost' identified by '5953';

select user,host from mysql.user;结果如图2.2



修改权限：

grant select,insert,update on courselab.\* to user1@localhost;

用户1对实验查看查找更新权利

grant select on \*.\* to user2@localhost;

用户2对数据库所有表查看权

刷新：

mysql>flush privileges;

对所有数据库实例权限写入了user表，对某个数据库权限写入了db表。

revoke删除用户权限，DROP USER 'username'@'host';删除用户，也可以直接操作user表 db表。配置权限后user和db表如图2.3和图2.4

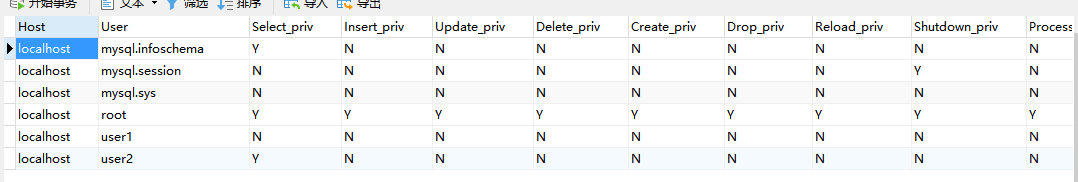


图2.3 user表

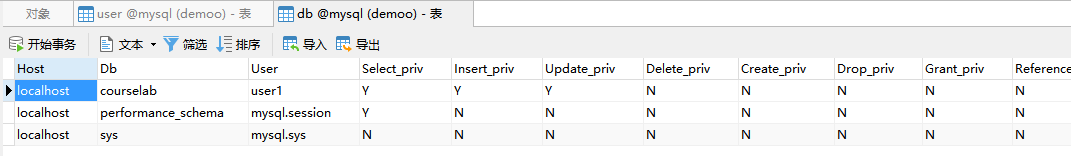


图2.4 db表

## 2.3任务总结

脱机备份过程按照网上的方法失败多次，导致mysql崩溃，原因是版本问题和个人的文件目录不同的问题，详细学习了目录结构，试了多次才解决问题。

# 3交互式使用SQL语言

## 3.1 任务要求

完成疫情前后火车营运及乘客乘车接触等表的创建，更新和查询操作。

## 3.2 完成过程

### 3.2.1 建表

1）创建下列跟“疫期乘坐列车”相关的关系，包括主码和外码的说明

a、车站表【车站编号，车站名，所属城市】

Station (SID int, SName char(20), CityName char(20))

其中，主码为车站编号。

b、车次表【列车流水号，发车日期，列车名称，起点站编号，终点站编号，开出时刻，终点时刻】

Train (TID int, SDate date, TName char(20), SStationID int, AStationID int, STime datetime, ATime datetime)

其中，TID为主码，(列车名称，发车日期)为候选码；SStationID和AStationID都来源于车站表的SID。

建立中间结果表，两层循环，取中间表的每行，从2019.12.2~2020.3.1安排时间insert到station表，候选码只是提示，可以不设置。

c、车程表【列车流水号，车站序号，车站编号，到达时刻，离开时刻】

TrainPass (TID int, SNo smallint, SID int, STime datetime, ATime datetime)

其中，主码为(TID,SNo)。SID来源于车站表的SID。

d、乘客表【乘客身份证号，姓名，性别，年龄】

Passenger (PCardID char(18), PName char(20), Sex bit, Age smallint)

其中，主码为乘客身份证号；性别取值为0/1（“1”表示“男”，“0”表示“女”）。

a、b、c、d表均可以由提供的xls文件和助教提供的txt文件，利用navigate的“导入表”功能实现，只需要手动设置字段名称、类型、长度、主键等信息，注意导入时的日期和分隔的字符格式，通过“设计表”功能实现外键参照设置。

以TrainPass表为例，设计表内容如图3.1和3.2所示

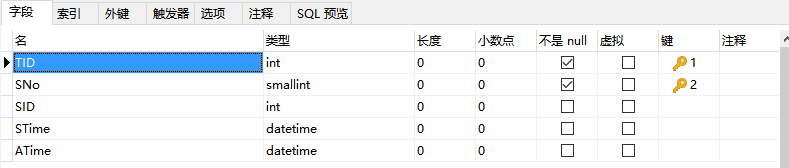
图3.1 TrainPass设计表字段、主键设置

图3.2 TrainPass设计表外键设置

e、乘车记录表【记录编号，乘客身份证号，列车流水号，出发站编号，到达站编号，车厢号，席位排号，席位编号，席位状态】

TakeTrainRecord (RID int, PCardID char(18), TID int, SStationID int, AStationID int, CarrigeID smallint, SeatRow smallint，SeatNo char(1)，SStatus int)

其中，主码、外码请依据应用背景合理定义。

CarrigeID若为空，则表示“无座”；

SeatNo只能取值为’A’、’B’、’C’、’E’、’F’，或为空值；

SStatus只能取值’0’（退票）、’1’（正常）、’2’（乘客没上车）。

采取SQl语句建表，为方便阅读，贴代码图如图3.3

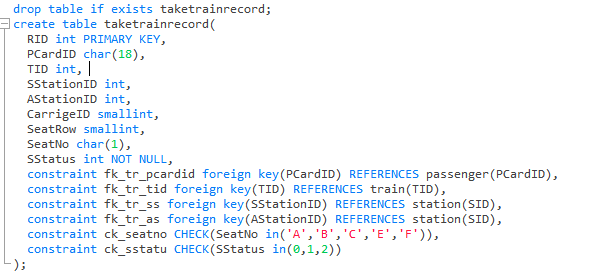


图3.3 TakeTrainRecord表建表sql语句

测试过程遇到问题1.mysql8.0.16后check语句起作用。2.在SeatNo未限制not null的情况下，SeatNo为null能通过 check(SeatNo in('A','B','C','E','F'))，把null写入check(SeatNo in('A','B','C','E','F') or SeatNo=null)反而测试出错。

填充乘车记录表，建立存储过程，利用游标，一层循环，从乘客表选取身份证号，1次乘车（0或1次后面查询很多没结果，改成乘车一次有所改善，改成0或多次会更好），随机从车次表选取tid，根据trainpass选取前后两个站作为上车站和下车站，随机车厢，座排，座位，同时检查座位冲突。用到了游标，random等函数。存储过程sql语句如图3.4



图3.4 填充乘车记录存储过程

执行call存储过程用时200s，建立记录25k

f、诊断表【诊断编号，病人身份证号，诊断日期，诊断结果，发病日期】

DiagnoseRecord (DID int, PCardID char(18), DDay date, DStatus smallint, FDay date)

其中，主码为DID；DStatus包括：1：新冠确诊；2：新冠疑似；3：排除新冠

建表语句如图3.5

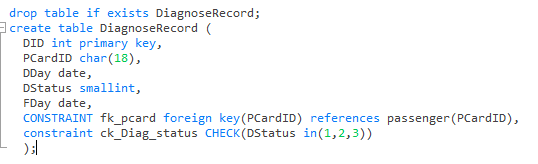


图3.5 诊断表建表

诊断表填充

模拟对乘客表中的每一位检测，检测时间2019.12.2~2020.3.1，对模拟为

确诊的患者模拟发病日期，诊断前4到14天。存储过程如图3.6



图3.6 填充诊断表存储过程

g、乘客紧密接触者表【接触日期, 被接触者身份证号，状态，病患身份证号】

TrainContactor (CDate date, CCardID char(18), DStatus smallint, PCardID char(18))

其中，主码为全码。DStatus包括：1：新冠确诊；2：新冠疑似；3：排除新冠

建表语句如图3.7

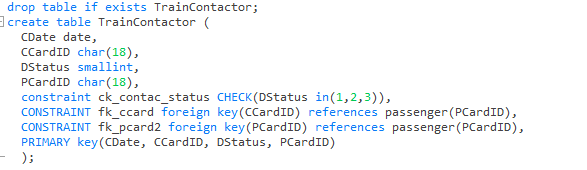
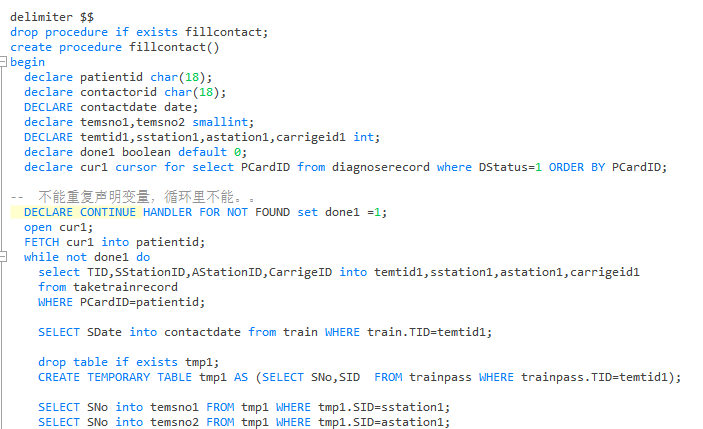
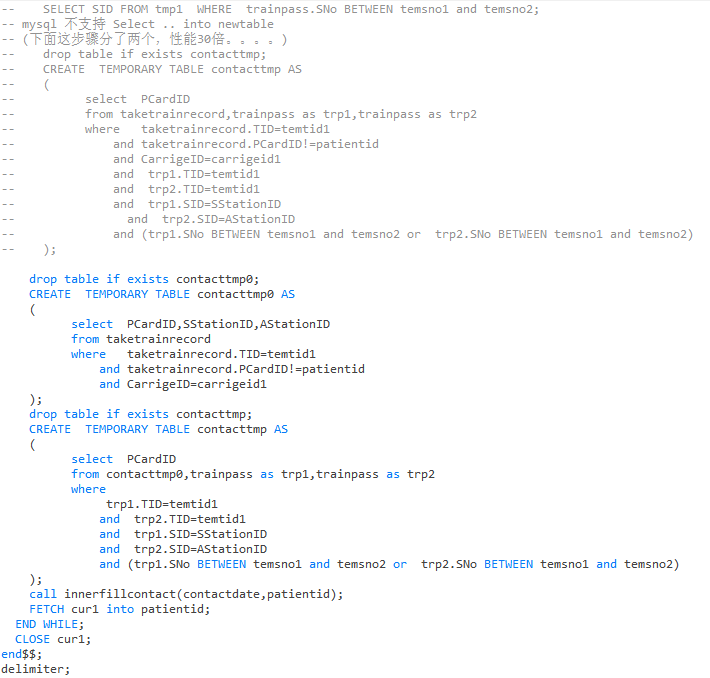


图3.7乘客紧密接触者建表sql

在诊断表中选取确诊乘客，根据乘车记录表找到其乘车的车次日期，找出站点区间同一车厢的乘客，为其模拟诊断状态，插入接触表。考虑到实际发病前也可能传染和为了简化实验，这里不再限制乘车日期要在发病日期后。遇到了双层游标，内层要用到外层结果，但是声明要在外层前，的问题。最后把内层变成了一个新的存储过程，调用。





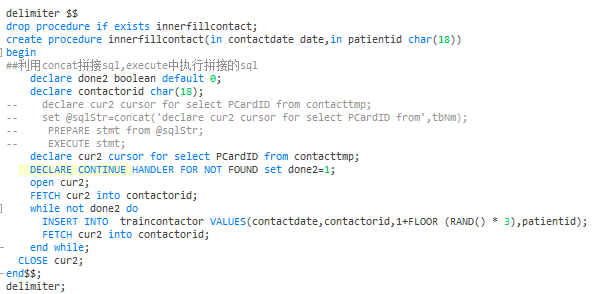


图3.8 填充密切接触存储过程语句

调用产生密切接触记录。

delete from traincontactor;

call fillcontact();

25k的诊断，只有几百个密切接触

验证结果，取一条记录，230231197505243594密切接触330803200111012166

查看乘车记录，发现正确。查询结果如图3.9

select \* from taketrainrecord WHERE PCardID='230231197505243594'

UNION

select \* from taketrainrecord WHERE PCardID='330803200111012166';

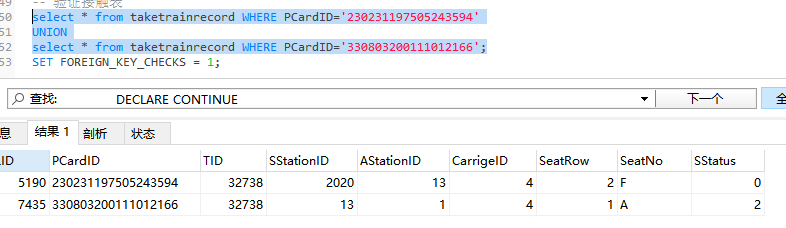


图3.9 密切接触测试

**2）观察性实验**

实验内容：验证在建立外码时是否一定要参考被参照关系的主码

观察用例1 参照被参照关系的非主码，结果出错，详细如图3.10

ALTER TABLE `trainpass` ADD FOREIGN KEY(`SID`) REFERENCES `train` (`SStationID`);

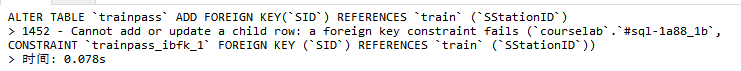


图3.10 被参照关系的非主码

观察用例2 参照被参照关系的主码，结果正常，详细如图3.11

ALTER TABLE `trainpass` ADD FOREIGN KEY(`SID`) REFERENCES `station` (`SID`);



图3.11 被参照关系的主码

### 3.2.2 更新

1）分别用一条sql语句完成对乘车记录表基本的增、删、改的操作；

操作语句如图3.12

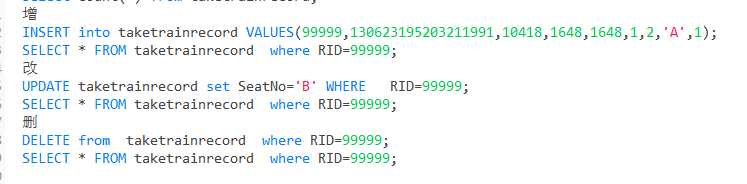


图3.12乘车记录表增删改

**2）批处理操作**

将乘车记录表中的从武汉出发的乘客的乘车记录插入到一个新表WH\_TakeTrainRecord中。

Mysql不支持select into from语句，利用creat table as 语句

如下图3.13

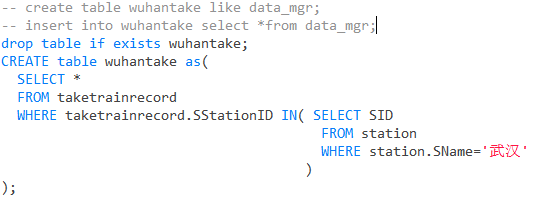


图3.13 武汉出发的乘客乘车记录

**3）数据导入导出**

通过查阅DBMS资料学习数据导入导出功能，并将任务2.1所建表格的数据导出到操作系统文件，然后再将这些文件的数据导入到相应空表。

navicate包含图形化的数据导入导出功能，可以设置xls，txt格式等。

**4）观察性实验**

建立一个关系，但是不设置主码，然后向该关系中插入重复元组，然后观察在图形化交互界面中对已有数据进行删除和修改时所发生的现象。

建立含有两列不设置主码的表，插入部分记录相等的数据进行删改，操作语句如图3.14

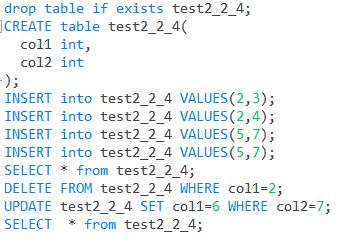


图3.14

删改后结果如图3.15

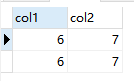


图3.15

由上图知符合选择条件的多个记录全被修改或删除。

**5）创建视图**

创建一个新冠确诊病人的乘火车记录视图，其中的属性包括：身份证号、姓名、年龄、乘坐列车编号、发车日期、车厢号，席位排号，席位编号。按身份证号升序排序，如果身份证号一致，按发车日期降序排序（注意，如果病人买了票但是没坐车，不计入在内）。

操作sql语句如图3.16

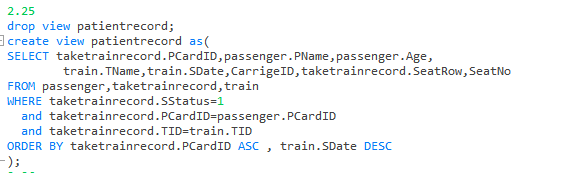


图3.16 新冠确诊病人的乘火车记录视图

**6）触发器实验**

编写一个触发器，用于实现以下完整性控制规则：

1) 当新增一个确诊患者时，若该患者在发病前14天内有乘车记录，则将其同排及前后排乘客自动加入“乘客紧密接触者表”，其中：接触日期为乘车日期。

2) 当一个紧密接触者被确诊为新冠时，从“乘客紧密接触者表”中修改他的状态为“1”。

mysql触发器不能用insert or update，这里仅仅insert

触发器建立语句如图3.17



图3.17 触发器建立

触发器测试，测试过程如图3.18

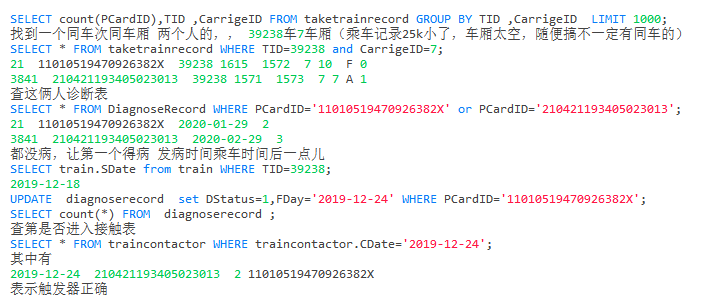


图3.18 触发器测试

### 3.2.3 查询

1）查询确诊者“张三”的在发病前14天内的乘车记录；

找出一个确诊患者，刘丹星，查发病前14天内的乘车记录。查询语句如图3.19

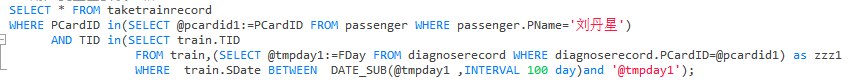


图3.19

2）查询所有从城市“武汉”出发的乘客乘列车所到达的城市名；

查询语句如图3.20

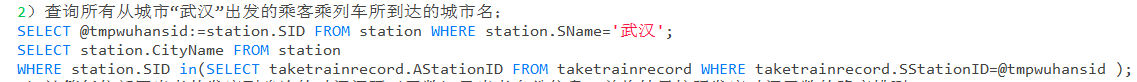


图3.20

3）计算每位新冠患者从发病到确诊的时间间隔（天数）及患者身份信息，并将结果按照发病时间天数的降序排列；如图3.21

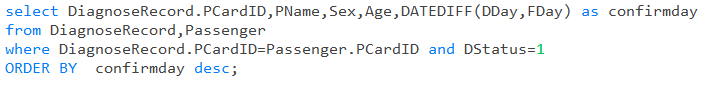


图3.21

4）查询“2020-01-22”从“武汉”发出的所有列车；如图3.22

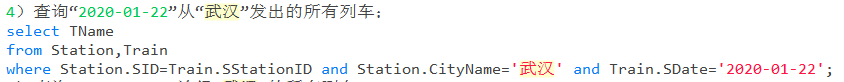


图3.22

5）查询“2020-01-22”途经“武汉”的所有列车；如图3.23

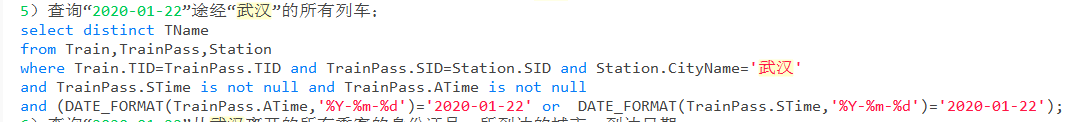


图3.23

6）查询“2020-01-22”从武汉离开的所有乘客的身份证号、所到达的城市、到达日期； 如图3.24

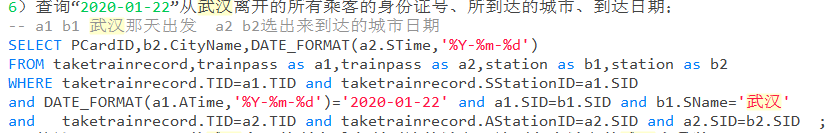


图3.24

7）统计“2020-01-22” 从武汉离开的所有乘客所到达的城市及达到各个城市的武汉人员数。如图3.25

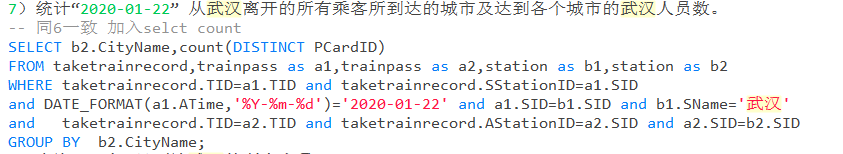


图3.25

8）查询2020年1月到达武汉的所有人员；如图3.26

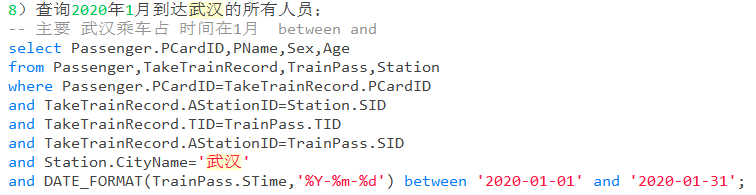


图3.26

9） 查询2020年1月乘车途径武汉的外地人员（身份证非“420”开头）；如图3.27

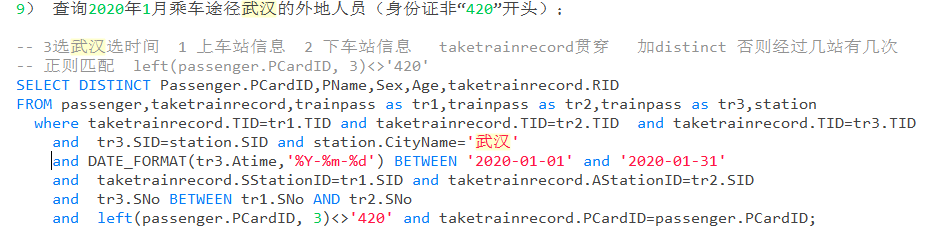


图3.27

10）统计“2020-01-22”乘坐过‘G007’号列车的新冠患者在火车上的密切接触乘客人数（每位新冠患者的同车厢人员都算同车密切接触）。如图3.28

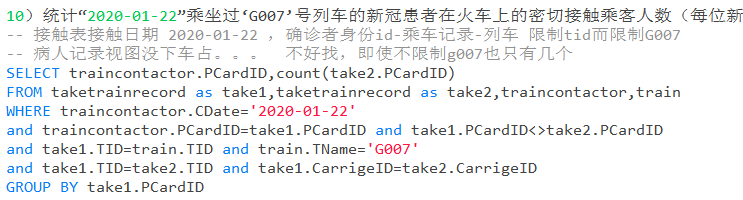


图3.28

11）查询一趟列车的一节车厢中有3人及以上乘客被确认患上新冠的列车名、出发日期，车厢号； 如图3.29

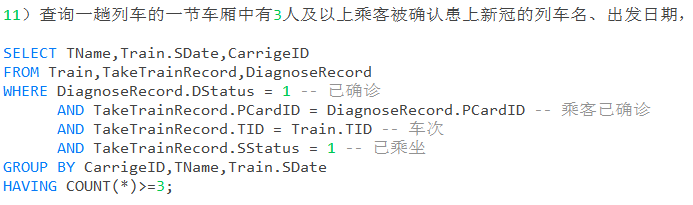


图3.29

12）查询没有感染任何周边乘客的新冠乘客的身份证号、姓名、乘车日期；如图3.30

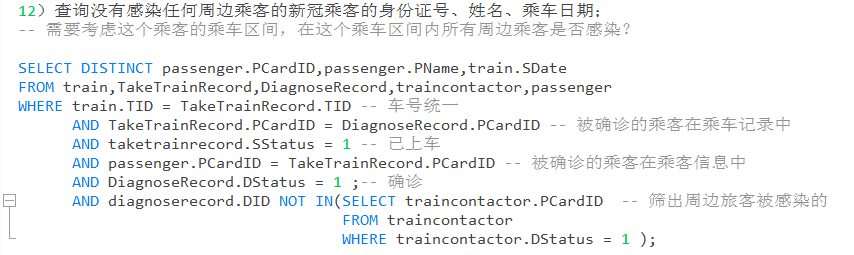


图3.30

13）查询到达 “北京”、或“上海”，或“广州”（即终点站）的列车名，要求where子句中除了连接条件只能有一个条件表达式；如图3.31

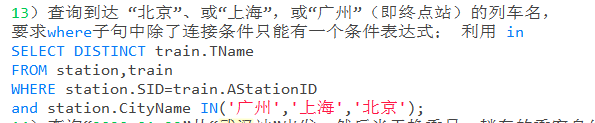


图3.31

14）查询“2020-01-22”从“武汉站”出发，然后当天换乘另一趟车的乘客身份证号和首乘车次号，结果按照首乘车次号降序排列，同车次则按照乘客身份证号升序排列；如图3.32

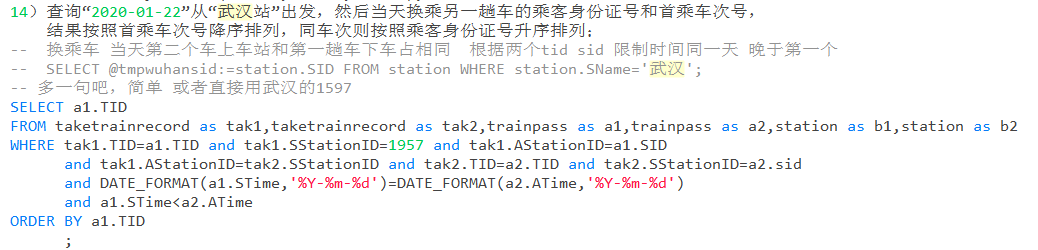


图3.32

15）查询所有新冠患者的身份证号，姓名及其2020年以来所乘坐过的列车名、发车日期，要求即使该患者未乘坐过任何列车也要列出来；如图3.33

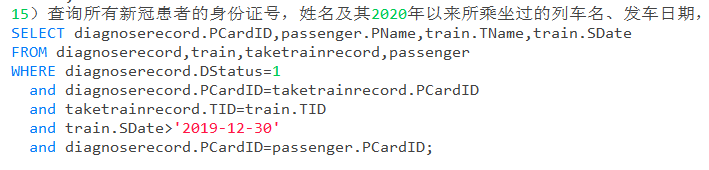


图3.33

16）查询所有发病日期相同而且确诊日期相同的病患统计信息，包括：发病日期、确诊日期和患者人数，结果按照发病日期降序排列的前提下再按照确诊日期降序排列。如图3.34

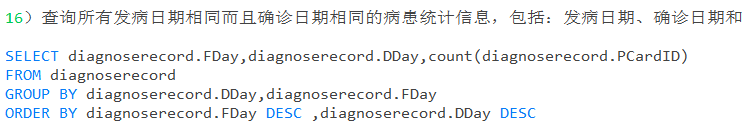


图3.34

## 3.3任务总结

总体感觉实验并不难，但是两周时间短，为了实验需要学mysql的语言格式，一些函数，这些比较简单，第一次接触存储函数有点难度，虽然没完成建立函数或存储过程的任务5但是前面实验大量用了存储过程。查询过程难度几种在7到12题，需要连接的表有点多，其实不必追求一句sql完成的话会更简单效率也更高，如实验反复用到了查武汉相关车次tid得反复嵌套查询。

还有碰到的问题是mysql并不完善，碰到一些错误，找不出错误，其实是mysql的问题，比如低版本check语句不报错，但是一点用没有；触发器不能 insert or update；更新版本、迁移数据文件、备份、等遇到的其它问题可见个人博客

<https://blog.csdn.net/qq_42544450/article/details/106132278>

https://blog.csdn.net/qq\_42544450/article/details/105938274

# 4 仓储管理系统

## 4.1系统设计目标

仓储管理系统是当今几乎所有商业管理中波不可缺的部分，常见的仓库的日常工作包括货品的入库和出库。入库要由采购人员提供进货单，进货单经过审核人员审核验收后方能进行货品入库。出库要由销售人员提供出货单，经过审核人员审核批准后才能提货。当销售人员需要提货而货品的库存量不足时可先进行缺货登记，当有相应货品入库时，按缺货登记时间顺序处理出货请求。

## 4.2 需求分析

能实现库存货品信息的管理，货品的相关信息包括：货品号、货品名、存放地、货品库存量、生产厂家等。

实现进货单的填写、修改、审核和查询等功能。

实现出货单的填写、修改、审核和查询等功能。

实现缺货登记、查询和处理。

根据不同用户身份提供不同的操作权限和界面。

完整性需求表现在职员，进出货，库存缺货相关的实体完整性和之间的参照完整性，以及用户对订单状态等自定的完整性约束，保证不同数据之间一致相容。

## 4.3 总体设计

整体目标为一个可视化管理应用，职工可根据自己的职位选择账户密码登录，

采购人员登录后可进行填写、修改、提交采购订单，和查询功能，查询项可限制订单状态，缺货信息，自己的订单统计；

审核人员登录后可进行审核采购单核销售，和查询功能，查询项可限制订单状态，缺货信息，自己的审核订单统计；

销售人员登录后可进行填写、修改、提交销售订单，和查询功能，查询项可限制订单状态，缺货信息，自己的订单统计；

## 4.4 数据库设计

依据题目设置如下数据表,索引,触发器

1、职员表【姓名，工号，职位】

工号为主码

职务包括 审核人员 销售人员 采购人员

2、账户表【账号（同工号），密码】

账号为主码，参考外键职员表工号

3、库存表【货号，货名，存放地，库存数量，生产厂家】

货号为主码

3、进货单【单号，货号，货名，下单时间，结单时间，完成状态，采购人工号，审核人工号】

单号为主码

货号外键参考库存表货号，采购人，审核人外键参考职员表。

完成状态 0，1，0代表待审核，1代表完成，完成的订单不可修改。

4、出货单【单号，货号，货名，下单时间，结单时间，完成状态，销售工号，审核人工号】

单号为主码

货号外键参考库存表货号，销售人，审核人外键参考职员表。

完成状态 0或1或2，0代表待审核，1代表完成，-1代表审核完成但是缺货。 完成的订单不可修改，当补货后达到所需出货数量时自动出货，完成状态变为1。

5缺货表【缺货货号，缺货数量】

货号为主键，参考库存表货号。

6 采购审核通过触发器，当采购被审核后，根据采购量，缺货量，库存量更新

采购单状态，库存，缺货表。

7 销售审核通过触发器，当销售单被审核后，根据销售量，缺货量，库存量更新

采销售状态，库存，缺货表。

8 其它为加速查询提供的视图。

由于采用图形界面实现，对不同职位提供不同的操作，不需要额外单独对用户设置权限。

## 4.5 详细设计与实现

实验采取java+jdbc+mysql+javafx实现带图形界面的BS应用。因为是数据库实验，java部分，GUI部分文字简要介绍，程序健壮性不再展示。

### 控制流设计

1. 场景界面功能简介

A 登录界面，账户密码登录，退出，注销

B 采购者界面，订单采购、查看采购单、退单、查看库存和缺货

C 销售者界面，订单销售、查看销售单、退单、查看库存和缺货

D 审核者界面，查看销售和采购订单，审核订单，查看库存和缺货

以上订单界面都有排序，限制只显示经本人办理订单，限制只显示当前订单功能。

2. 账户登录

职工通过账户密码登录系统，其中账号密码由数据库中account表提供检查。

账户格式为 字母C/X/S+5位数字，C代表采购员，X销售员，S审核员，

当账户密码匹配时，通过账户首字母导向不同职员的场景界面。

注销和退出，当点击注销时退出登录，点击退出时，保存更改，退出系统，下面每个界面都有这两项功能，不再叙述。

3. 采购者界面

通过查询库存表、缺货表、采购表加载数据到相应商品选择框或展示表格。

考虑现实需求，采购通过选择货品名字和输入数量订购，过程中自动匹配货品的编号，通过得到的标号和数量生成 订单状态为0（下单未审核）的订单，

其中下单时间，下单职员自动填写。审核员，结单时间置空。更新采购表。

查看采购单界面可选择状态为0（未审核）的订单退单，由得到的订单号更新采购表。

当选择隐藏历史订单、只显示经本人办理的订单时，再次通过where条件设置查询采购表，更新表格内容。

4. 销售者界面

通过查询库存表、缺货表、销售表加载数据到相应商品选择框或展示表格。

考虑现实需求，采购通过选择货品名字和输入数量订购，过程中自动匹配货品的编号，通过得到的标号和数量生成 订单状态为0（下单未审核）的订单，

其中下单时间，下单职员自动填写。审核员，结单时间置空。更新销售表。

查看采购单界面可选择状态为0（未审核）的订单退单，由得到的订单号更新采购表。

5. 审核者界面

通过查询库存表、缺货表、采购表、销售表加载数据到相应展示表格。

查看采购或者销售订单，对其中订单状态为0的订单可以选择审核。审核完成后更新销售/采购表订单状态为1，结单时间，审核人，计算库存量和缺货量没更新库存表，缺货表。审核只需要将状态改为1，已审核，是否缺货及缺货补货由更新操作触发的触发器处理。

### 数据库详细设计

具有代表性的表，事件如下

1.库存表如图4.1

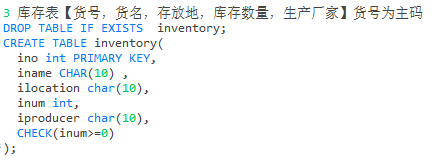


图4.1

2.销售表如图4.2

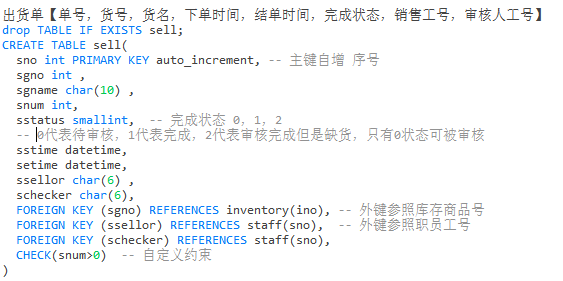


图4.2

3. 采购触发器如图4.3

采购通过审核后。判断是否缺货，处理库存和缺货，逻辑见注释



图4.3

4.销售单被审核触发器如图4.4

销售触发 审核后销售订单状态变1 ，如果货足，发货库存减，否则订单状态改为2，填缺货表。

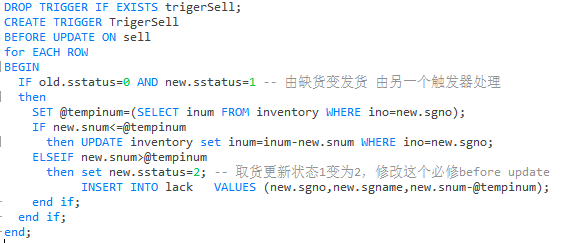


图4.4

## 4.6 系统测试

由于是数据库实验，仅展示主要数据库相关内容，错误处理，账户控制、登录注销，界面等不单独展示，在功能中顺带展示。

### 1、系统主界面

如图4.5



图4.5

### 2、测试过程简介

展示数据表原本内容，这里展示库存表、销售表，采购表，缺货表（当前为空，略）。

登录销售员x00002，展示他能看到的库存信息，展示他能看到的销售订单。

由于采购，销售，审核查库存和缺货的功能完全一致，销售，采购，审核查订单极度相似，区别仅仅是销售查sell表，采购查buy表，审核查两个表union，sell、buy表结构又一致。略。

测试订单查询的额外功能，为每种职员查询订单都提供了，隐藏历史订单、即仅显示当前未处理结束订单、仅仅显示经本人办理的订单、按某一列排序。当然也可以同时限制。不同职员区别是以各自工号作为查询where条件，这里仅以销售x00002展示。

限制查询过程，页面上虽然有刷新按钮，但是已经实现自动刷新。

以销售员x00002测试订单删除，同理采购订单删除略。

以销售x00002 展示商品销售页面，同理采购页面略。

**重点**：分别以销售x00002 ，采购c00002，审核s00001登录，进行系列采购销售审核。同时体现库存和缺货的变化，具体是如下

A、销售选择某一商品销售不导致缺货的数量。

B、销售选择某一商品销售会导致缺货的数量。

C、查看订单，预期销售订单多出两个相应商品的销售单，状态为0（未审核），自动填写了下单时间，下单职员工号x00002。

D、注销，以采购c00002登录、采购某一不缺货商品

E、采购某一已经缺货商品（过程B中商品）但是数量依旧不足。

F、采购某一已经缺货商品（过程B中商品）数量足以弥补缺货。

G、注销，以审核s00001登录、到目前为止、因为没审核，库存缺货不变。

H、审核过程A的销售单，预期库存减少,该订单状态改为1（ 已完成）。同时补足订单的结单时间，审核人工号。往后的补足结单时间、审核人不再展示。

I、审核过程B的销售单，预计订单状态变为2（已审核，缺货待补货），库存不变，缺货表增加记录。

J、审核过程D中的采购单，预期库存增加。

K、审核过程E中的采购单，预期库存增加，相应缺货数减少。

L、审核过程F中的采购单，预期库存增加，缺货记录删除，同时B销售单状态由2变为1，代表货补够了，真正售出。

至此完成了核心功能的代表性测试。再次展示数据表，以库存表为例，证明ui中的表操作与数据库实际更改一致。

### 3、具体测试及截图

库存表如图4.6

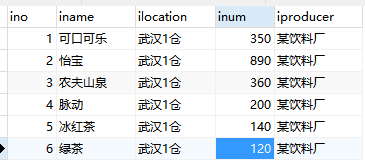


图4.6

销售表如图4.7

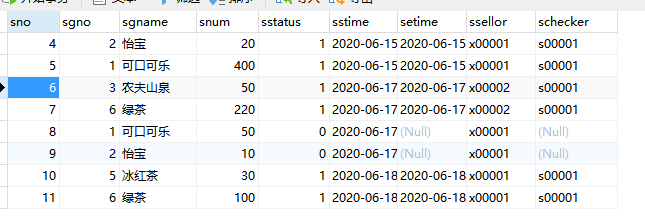


图4.7

采购表如图4.8

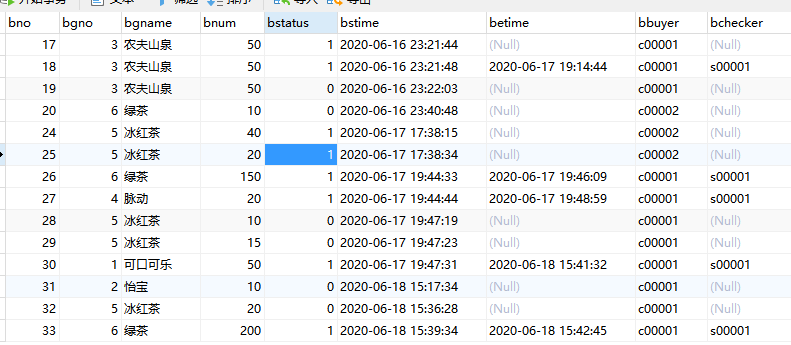


图4.8

销售登录后能看到的库存表如图4.9

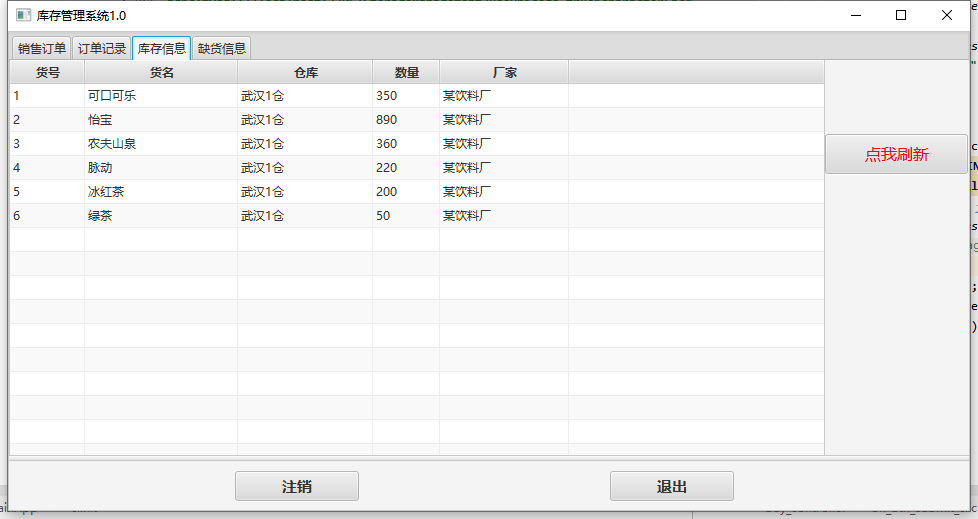
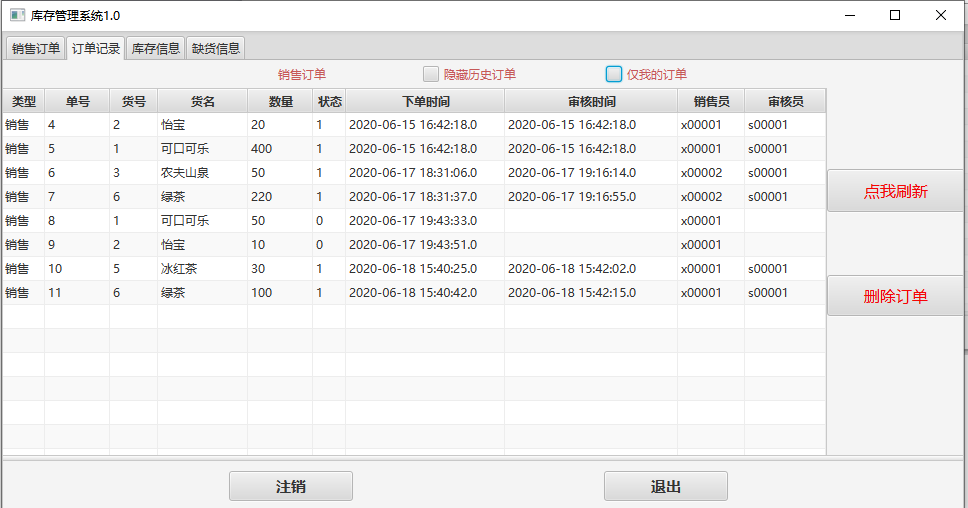


图4.9

同库存一致。

销售登录后能看到的销售表如图4.10



同销售订单表一致。

图4.10

销售x00002登录后限制了隐藏历史订单的表如图4.11

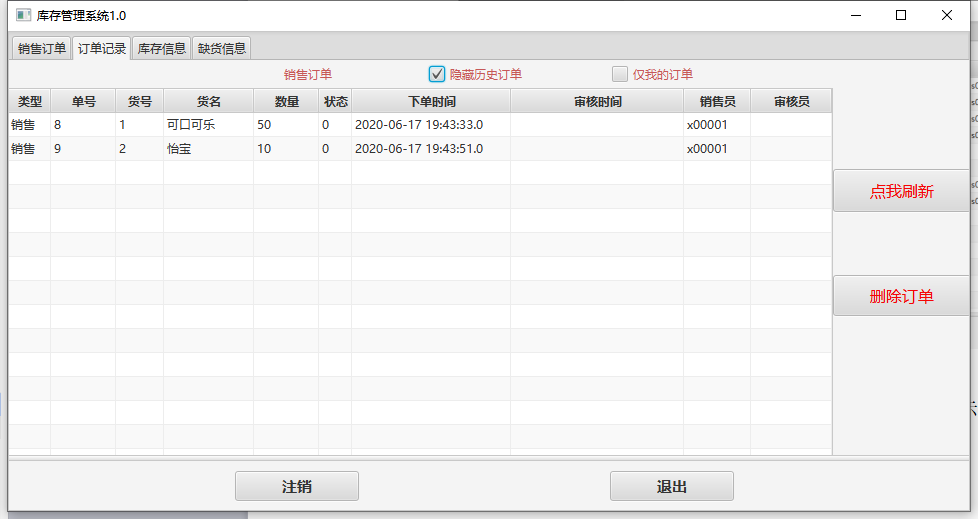


图4.11

仅剩下未被审核的订单。

销售x00002登录后限制了仅我的订单的表如图4.12

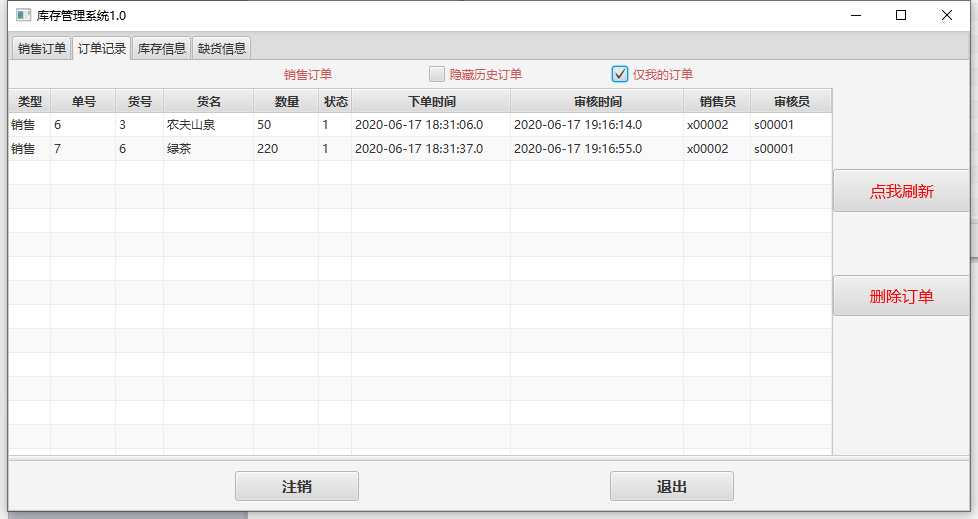


图4.12

仅剩下我办理的订单。

销售x00002进行订单销售，如图4.13

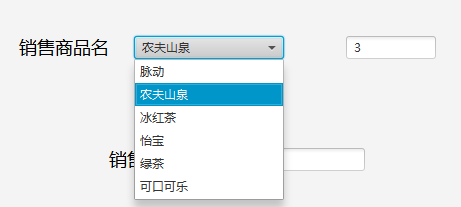


图4.13

可销售商品由查询仓库得出商品种类，并依照现实习惯按名字选取，商品号为查询自动匹配。

A、销售选择某一商品销售不导致缺货的数量。提交，如图4.14

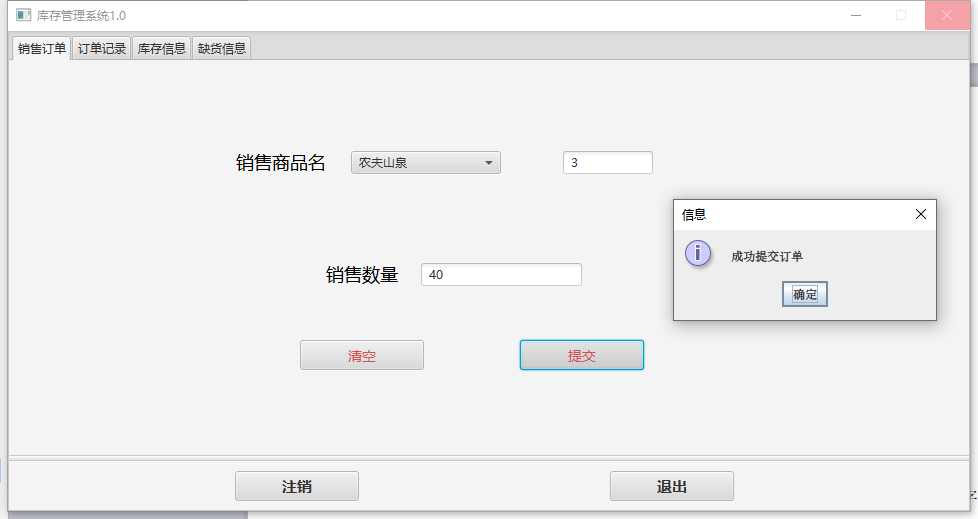


图4.14

B、销售选择某一商品销售会导致缺货的数量。提交，如图4.15

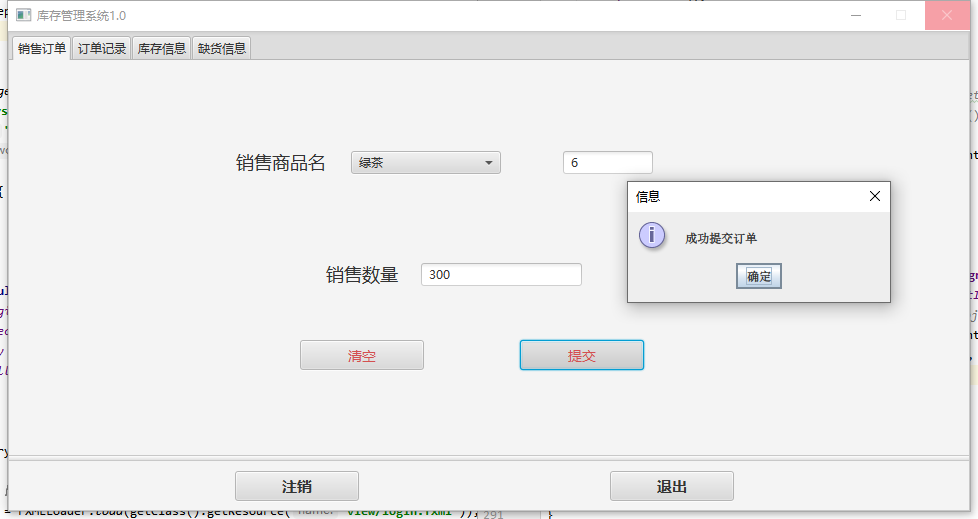


图4.15

C、查看订单，销售订单前两行两个相应商品的销售单（同时选择下单时间倒序排列），状态为0（未审核），自动填写了下单时间，下单职员工号x00002。如图4.16



图4.16

限制功能，隐藏历史订单如图4.17，剩下都是未完成的。

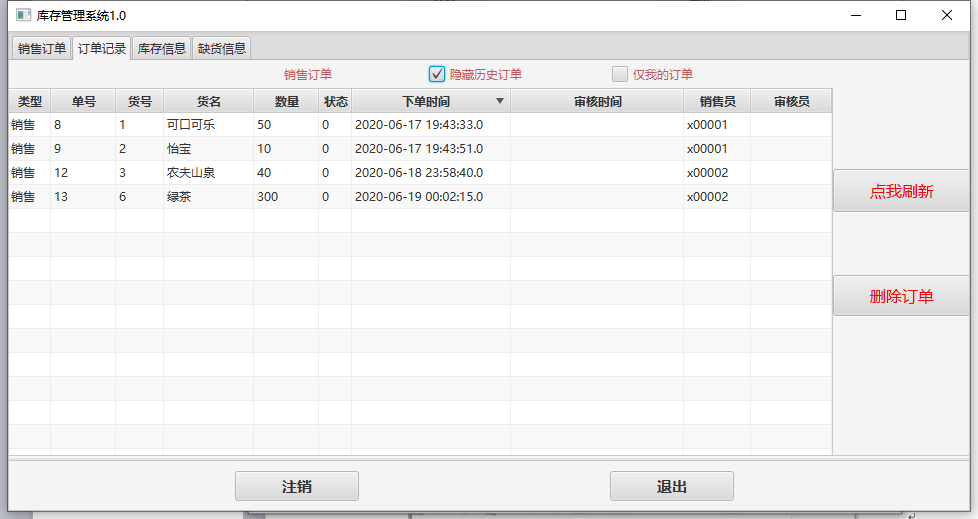


图4.17

限制功能，同时仅我的订单如图4.18，剩下都是我提交的未审核的。

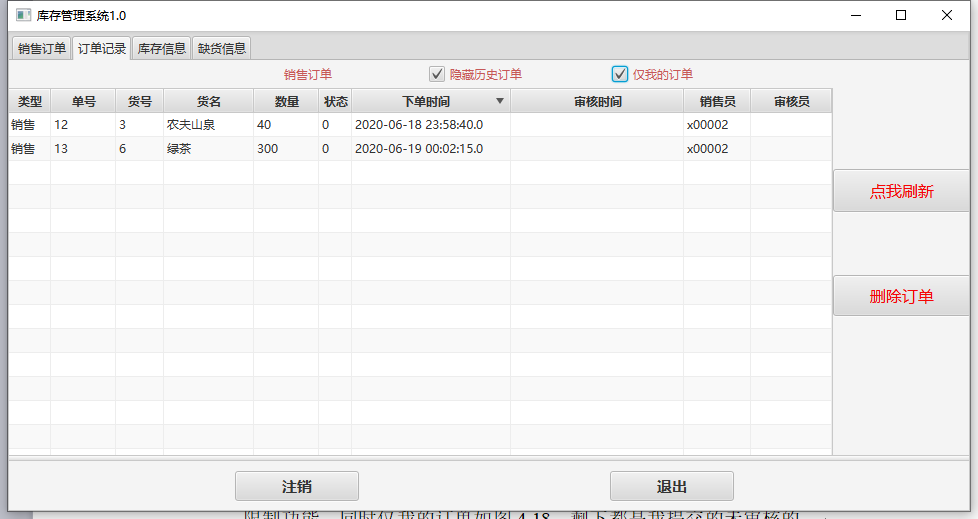


图4.18

D、注销，以采购c00002登录、采购某一不缺货商品。

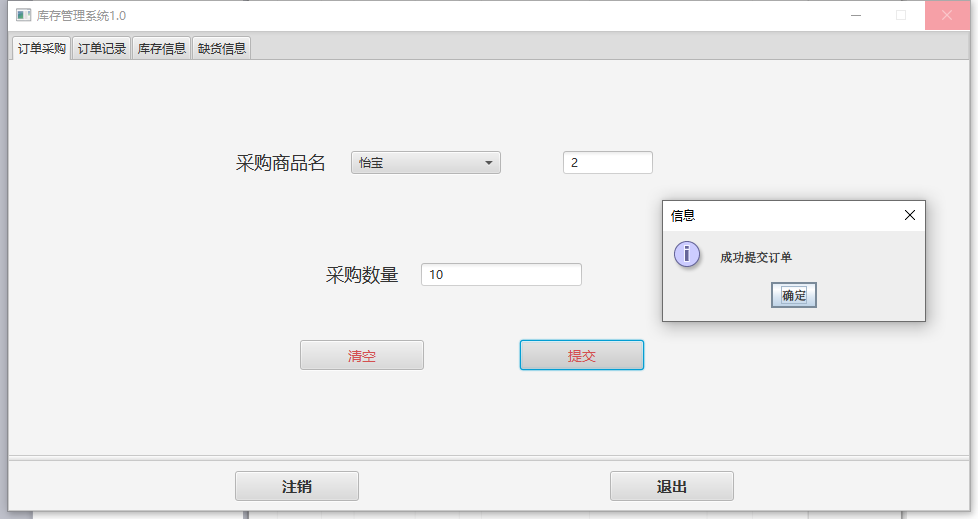


图4.19

E、采购某一已经缺货商品（过程B中商品）但是数量依旧不足。

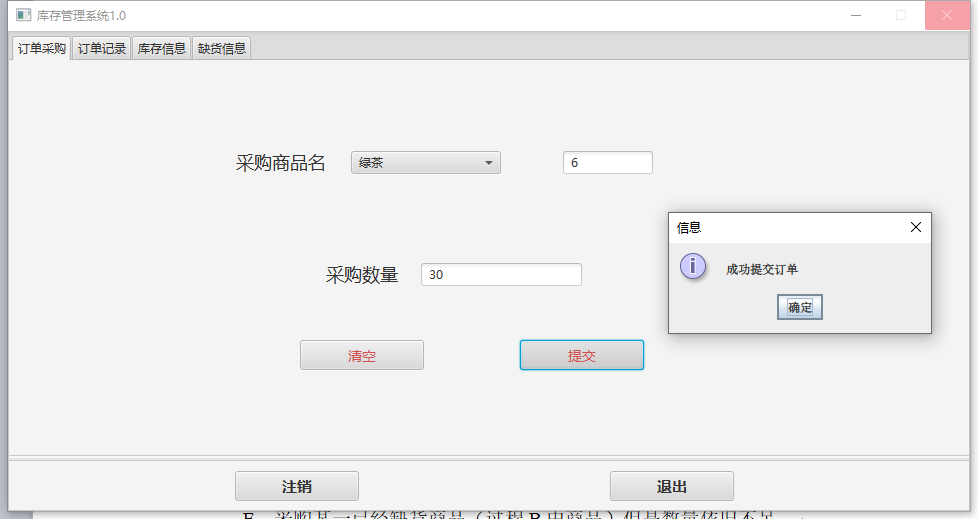


图4.20

F、采购某一已经缺货商品（过程B中商品）数量足以弥补缺货。

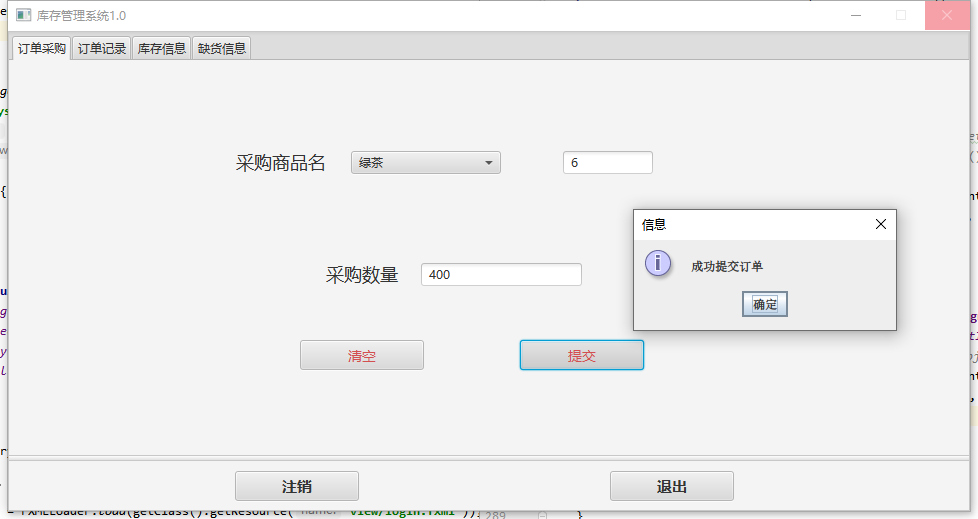


图4.21

G、注销，以审核s00001登录、到目前为止、因为没审核，库存缺货不变。展示审核人看到的的订单（已经排序）。前五行即新单。最左边是类型。



图4.22

H、审核过程A的销售单（第五行）。

该订单状态改为1（ 已完成）。同时补足订单的结单时间，审核人工号。

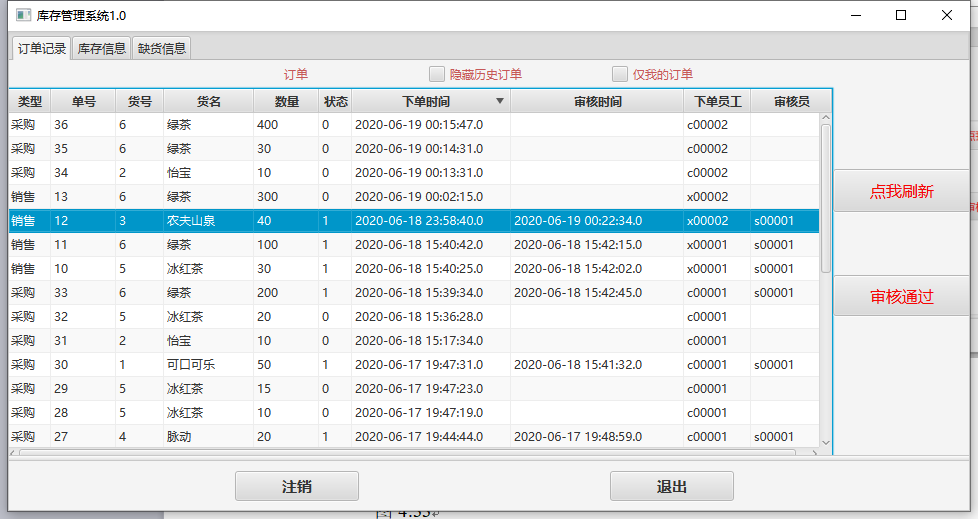


图4.23

库存由前面图360减少到320,如图4.24

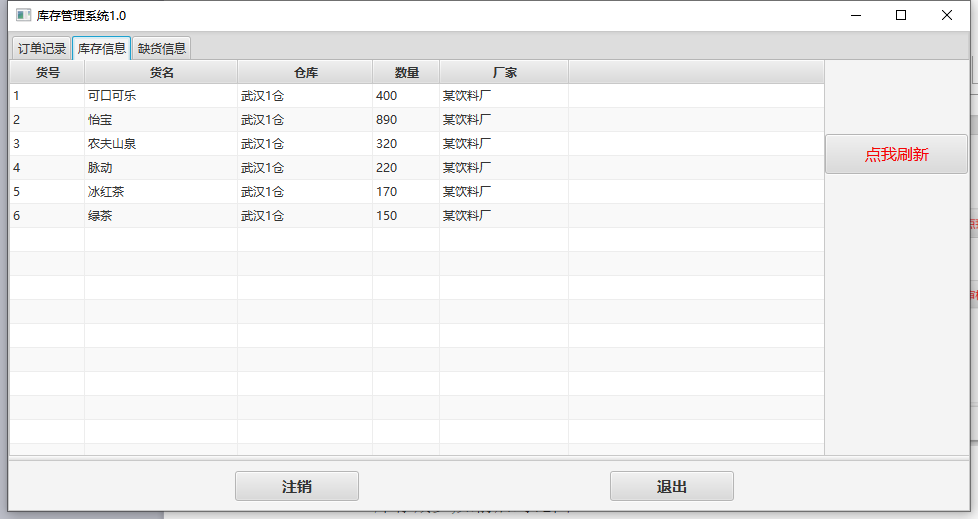


图4.24

I、审核过程B的销售单，

订单状态变为2（已审核，缺货待补货）。



图4.25

库存不变，缺货表增加记录。库存150，要销售300，缺150

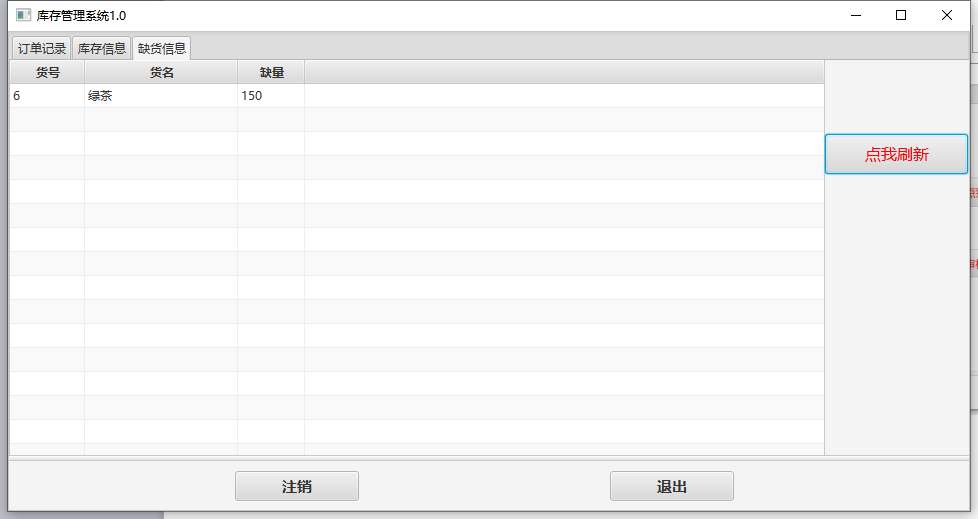


图4.26

J、审核过程D中的采购单，库存增加。

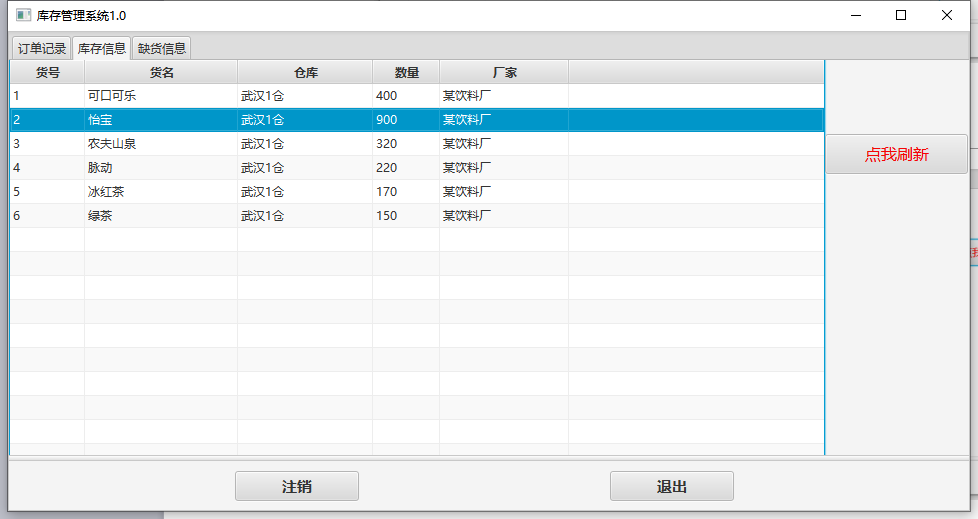


图4.27

K、审核过程E中的采购单，库存增加，相应缺货数减少。

采购30绿茶，库存数150->180



图4.28

缺货数150->120



图4.29

L、审核过程F中的采购单。采购400绿茶

库存增加， 400-120=280



图4.30

缺货记录删除。

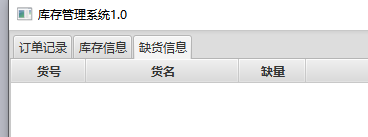


图4.31

同时B销售单状态由2变为1，代表货补够了，真正售出。



图4.32

至此完成了核心功能的代表性测试。再次展示数据库数据表，以库存表为例，证明ui中的表操作与数据库实际更改一致。如图4.33

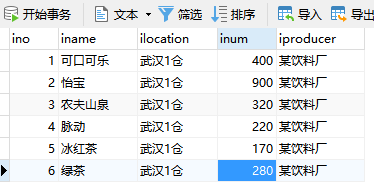


图4.33

## 4.7 系统设计与实现总结

分条目概括、总结应用系统设计的主要工作。

1、分析需求、设计数据表及主外键约束，能完整存下系统所依赖或存储的数据。

2、优化表结构、自定义约束，增加效率确保数据正确完整

3、当数据量较大时应该设计视图，索引加快查询。也可以通过视图间接限制用户权限。建立触发器，及时处理相关联时间，保证数据一致相容。同时触发器能减少控制程序中数据库相关操作，如实验中不同职工实际上都只能查询库存和缺货，删除更新操作完全由触发器实现。

4、本实验中因为要为不同类型职工提供不同操作界面，需要认真设计控制流程。因为B-S系统设计目标是服务商业，控制流程的出错处理也十分重要，保证程序健壮是工程师的基本责任。

# 5 课程总结

逐条概括、总结此次课程实践的主要工作，阐述此次课程实践的心得体会，展望此次课程实践的有待改进和完善的工作。

1、 通过过学习navicate软件了解通过软件爱你建表、视图、触发器等操作和数据库的软件备份方法。

2、 通过学习mysql的文件结构和日志记录学到了脱机备份与恢复数据库的方法。

3、了解了数据库增删用户及权限的方法和 直接管理mysql中与之先关的user表和db表的操作方法。

4、增加了sql建表,操作数据记录，建立视图，差法器等操作等熟练度、加深了触发器触发时间，视图加速查询的理解。

5、掌握了一定的存储过程和函数的能力，这方面实验给的资料很少，完成费时。

6、在完成实验1的过程中，虽然mysql能自己优化查询，但是自己用不同方式查询时间还是有明显差距，按照后面学的查询优化思路查询效率更高，且实验1显示只用1条sql语句查询，增加难度降低效率，虽然确实增加了对各种jion操作，复合语句的理解。

7、综合设计过程联系了其他课程，相互促进了不同课程知识的掌握，基本实现了一个功能健全的小型数据库应用。

8、课程实践的时间设置不合理，由于课程缺乏实践，实验1,2相比实验3只不过不需要自己设计，熟悉一个mysql语言，mysql文件结构、及学习存储过程，函数、等耗时两周时间不够，虽然老师允许了拖延。

# 6 附录

附录仅展示二 三 四 部分的sql代码，系统设计中的java源码在实验包中查看。

## 疫情火车sql

SET NAMES utf8mb4;

SET FOREIGN\_KEY\_CHECKS = 0;

-- ----------------------------

-- Table structure for passenger

-- ----------------------------

CREATE TABLE `passenger` (

`PCardID` char(18) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,

`PName` char(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL,

`Sex` tinyint(1) NULL DEFAULT NULL,

`Age` smallint(0) NULL DEFAULT NULL,

PRIMARY KEY (`PCardID`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci ROW\_FORMAT = Dynamic;

-- ----------------------------

-- Table structure for station

-- ----------------------------

CREATE TABLE `station` (

`SID` int(0) NOT NULL,

`SName` char(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL,

`CityName` char(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL,

PRIMARY KEY (`SID`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci ROW\_FORMAT = Dynamic;

-- ----------------------------

-- Table structure for train

-- ----------------------------

CREATE TABLE `train` (

`TID` int(0) NOT NULL,

`SDate` date NULL DEFAULT NULL,

`TName` char(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL,

`SStationID` int(0) NULL DEFAULT NULL,

`AStationID` int(0) NULL DEFAULT NULL,

`STime` datetime(0) NULL DEFAULT NULL,

`ATime` datetime(0) NULL DEFAULT NULL,

PRIMARY KEY (`TID`) USING BTREE,

INDEX `trainfk1`(`SStationID`) USING BTREE,

INDEX `trainfk2`(`AStationID`) USING BTREE,

CONSTRAINT `trainfk1` FOREIGN KEY (`SStationID`) REFERENCES `station` (`SID`) ON DELETE RESTRICT ON UPDATE RESTRICT,

CONSTRAINT `trainfk2` FOREIGN KEY (`AStationID`) REFERENCES `station` (`SID`) ON DELETE RESTRICT ON UPDATE RESTRICT

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci ROW\_FORMAT = Dynamic;

-- 验证在建立外码时是否一定要参考被参照关系的主码

-- ALTER TABLE `trainpass` ADD FOREIGN KEY(`SID`) REFERENCES `train` (`SStationID`);

ALTER TABLE `trainpass` ADD FOREIGN KEY(`SID`) REFERENCES `station` (`SID`);

-- ----------------------------

-- Table structure for trainpass

-- ----------------------------

CREATE TABLE `trainpass` (

`TID` int(0) NOT NULL,

`SNo` smallint(0) NOT NULL,

`SID` int(0) NULL DEFAULT NULL,

`STime` datetime(0) NULL DEFAULT NULL,

`ATime` datetime(0) NULL DEFAULT NULL,

PRIMARY KEY (`TID`, `SNo`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8\_general\_ci ROW\_FORMAT = Dynamic;

-- 乘车记录表【记录编号，乘客身份证号，列车流水号，出发站编号，到达站编号，车厢号，席位排号，席位编号，席位状态】

-- TakeTrainRecord (RID int, PCardID char(18), TID int, SStationID int, AStationID int, CarrigeID smallint, SeatRow smallint，SeatNo char(1)，SStatus int)

-- 其中，主码、外码请依据应用背景合理定义。

-- CarrigeID若为空，则表示“无座”；

-- SeatNo只能取值为’A’、’B’、’C’、’E’、’F’，或为空值；

-- SStatus只能取值’0’（退票）、’1’（正常）、’2’（乘客没上车）。

--

drop table if exists taketrainrecord;

create table taketrainrecord(

RID int PRIMARY KEY,

PCardID char(18),

TID int,

SStationID int,

AStationID int,

CarrigeID smallint,

SeatRow smallint,

SeatNo char(1),

SStatus int NOT NULL,

constraint fk\_tr\_pcardid foreign key(PCardID) REFERENCES passenger(PCardID),

constraint fk\_tr\_tid foreign key(TID) REFERENCES train(TID),

constraint fk\_tr\_ss foreign key(SStationID) REFERENCES station(SID),

constraint fk\_tr\_as foreign key(AStationID) REFERENCES station(SID),

constraint ck\_seatno CHECK(SeatNo in('A','B','C','E','F')),

constraint ck\_sstatu CHECK(SStatus in(0,1,2))

);

-- --test

-- INSERT INTO `taketrainrecord` VALUES (1, '110101193412052028',1,1599,1621,18,5,'A',1);

-- INSERT INTO `taketrainrecord` VALUES (2, '110101193412052028',1,1599,1621,18,5,NULL,1);

-- INSERT INTO `taketrainrecord` VALUES (4, '110101193412052028',1,1599,1621,18,5,'K',1);

-- INSERT INTO `taketrainrecord` VALUES (3, '110101193412052028',1,1599,1621,18,5,'A',3);

-- ;

--

-- 诊断表【诊断编号，病人身份证号，诊断日期，诊断结果，发病日期】

-- DiagnoseRecord (DID int, PCardID char(18), DDay date, DStatus smallint, FDay date)

-- 其中，主码为DID；DStatus包括：1：新冠确诊；2：新冠疑似；3：排除新冠

drop table if exists DiagnoseRecord;

create table DiagnoseRecord (

DID int primary key,

PCardID char(18),

DDay date,

DStatus smallint,

FDay date,

CONSTRAINT fk\_pcard foreign key(PCardID) references passenger(PCardID),

constraint ck\_Diag\_status CHECK(DStatus in(1,2,3))

);

-- select concat( CONCAT(FLOOR(2015 + (RAND() \* 1)),'-',LPAD(FLOOR(10 + (RAND() \* 2)),2,0),'-',LPAD(FLOOR(1 + (RAND() \* 25)),2,0)))

-- 结果：2015-11-15

-- select CONCAT(LPAD(FLOOR(0 + (RAND() \* 23)),2,0),':',LPAD(FLOOR(0 + (RAND() \* 59)),2,0),':',LPAD(FLOOR(0 + (RAND() \* 59)),2,0))

-- 结果：20:57:10

delimiter $$

drop procedure if exists filldiag;

create procedure filldiag()

begin

declare PCardid1 char(18);

declare didcount int DEFAULT 0;

declare dstatus1 int DEFAULT 3;

DECLARE diagdate,fdate,begindate date;

declare done1 boolean default 0;

declare cur1 cursor for select PCardID from passenger ORDER BY PCardID;

DECLARE CONTINUE HANDLER FOR NOT FOUND set done1 =1;

open cur1;

FETCH cur1 into PCardid1;

while not done1 do

set begindate='2019-12-02';

-- 设置时间差，开始+三个月随机时间差=诊断日 如果确诊，诊断日- 0到14 发病日期

set diagdate=date\_add(begindate,interval 0+FLOOR(RAND() \* 90) day);

set dstatus1=FLOOR(1+ (RAND() \* 3));

-- 诊断状态 1 2 3

if dstatus1!=1

then set fdate=null;

else set fdate= DATE\_SUB(diagdate, interval 4+FLOOR (RAND() \* 10) day);

-- 4到14天前发病

end if;

-- select char(if(floor(rand()\*2)=0,65+floor(rand()\*26),48+floor(rand()\*9))); 随机日期

set didcount=didcount+1;

INSERT INTO DiagnoseRecord VALUES(didcount,PCardid1,diagdate,dstatus1,fdate);

FETCH cur1 into PCardid1;

END WHILE;

CLOSE cur1;

end$$;

delimiter;

delete from DiagnoseRecord;

call filldiag();

-- 乘客紧密接触者表【接触日期, 被接触者身份证号，状态，病患身份证号】

-- TrainContactor (CDate date, CCardID char(18), DStatus smallint, PCardID char(18))

-- 其中，主码为全码。DStatus包括：1：新冠确诊；2：新冠疑似；3：排除新冠

drop table if exists TrainContactor;

create table TrainContactor (

CDate date,

CCardID char(18),

DStatus smallint,

PCardID char(18),

constraint ck\_contac\_status CHECK(DStatus in(1,2,3)),

CONSTRAINT fk\_ccard foreign key(CCardID) references passenger(PCardID),

CONSTRAINT fk\_pcard2 foreign key(PCardID) references passenger(PCardID),

CONSTRAINT pk\_contactor PRIMARY key(CDate, CCardID, DStatus, PCardID)

);

ALTER TABLE traincontactor drop PRIMARY KEY;

ALTER TABLE traincontactor ADD CONSTRAINT pk\_contactor PRIMARY key(CDate, CCardID, DStatus, PCardID);

ALTER TABLE

-- -- 遇到某个游标的筛选条件来自于 先前语句运行的结果，比较常见的方式是 再写一个存储过程，通过调用来完成 动态参数的配置，或者使用 动态sql的功能

-- -- 接触表

-- 嵌套游标，内层用到了外层结果，但是游标不能再内层声明。

-- delimiter $$

-- drop procedure if exists fillcontact;

-- create procedure fillcontact()

-- begin

-- declare patientid char(18);

-- declare contactorid char(18);

-- DECLARE contactdate date;

-- declare temsno1,temsno2 smallint;

-- DECLARE temtid1,sstation1,astation1,carrigeid1 int;

-- declare done1 boolean default 0;

-- declare cur1 cursor for select PCardID from DiagnoseRecord where DStatus=1 ORDER BY PCardID;

-- declare done2 boolean default 0;

-- declare cur2 cursor for select PCardID from contacttmp;

-- -- 不能重复声明变量，循环里不能。。

-- DECLARE CONTINUE HANDLER FOR NOT FOUND

-- begin

-- -- 到了尾部

-- set done2 =1;

-- set done1 =1;

-- END;

--

-- open cur1;

-- FETCH cur1 into patientid;

-- while not done1 do

-- set done2=0;

-- select TID,SStationID,AStationID,CarrigeID into temtid1,sstation1,astation1,carrigeid1

-- from taketrainrecord

-- WHERE PCardID=patientid;

--

-- SELECT SDate into contactdate from train WHERE train.TID=temtid1;

--

-- drop table if exists tmp1;

-- CREATE TEMPORARY TABLE tmp1 AS (SELECT SNo,SID FROM trainpass WHERE trainpass.TID=temtid1);

--

-- SELECT SNo into temsno1 FROM tmp1 WHERE tmp1.SID=sstation1;

-- SELECT SNo into temsno2 FROM tmp1 WHERE tmp1.SID=astation1;

-- -- SELECT SID FROM tmp1 WHERE trainpass.SNo BETWEEN temsno1 and temsno2;

-- -- mysql 不支持 Select .. into newtable

-- drop table if exists contacttmp;

-- CREATE TEMPORARY TABLE contacttmp AS

-- ( select PCardID

-- from taketrainrecord

-- where TID=temtid1

-- and CarrigeID=carrigeid1

-- and (SStationID BETWEEN temsno1 and temsno2 or AStationID BETWEEN temsno1 and temsno2));

--

-- open cur2;

-- FETCH cur2 into contactorid;

-- while not done2 do

-- INSERT INTO VALUES(contactdate,contactorid,1+FLOOR (RAND() \* 3),patientid);

-- FETCH cur2 into contactorid;

-- end while;

-- set done1=0;

-- -- 文件尾外循环连带设置1了，改回

-- FETCH cur1 into patientid;

-- END WHILE;

-- CLOSE cur1;

-- end$$;

-- delimiter;

--

-- delete from traincontactor;

-- call fillcontact();

delimiter $$

drop procedure if exists innerfillcontact;

create procedure innerfillcontact(in contactdate date,in patientid char(18))

begin

##利用concat拼接sql,execute中执行拼接的sql

declare done2 boolean default 0;

declare contactorid char(18);

-- declare cur2 cursor for select PCardID from contacttmp;

-- set @sqlStr=concat('declare cur2 cursor for select PCardID from',tbNm);

-- PREPARE stmt from @sqlStr;

-- EXECUTE stmt;

declare cur2 cursor for select PCardID from contacttmp;

DECLARE CONTINUE HANDLER FOR NOT FOUND set done2=1;

open cur2;

FETCH cur2 into contactorid;

while not done2 do

INSERT INTO traincontactor VALUES(contactdate,contactorid,1+FLOOR (RAND() \* 3),patientid);

FETCH cur2 into contactorid;

end while;

CLOSE cur2;

end$$;

delimiter;

delimiter $$

drop procedure if exists fillcontact;

create procedure fillcontact()

begin

declare patientid char(18);

declare contactorid char(18);

DECLARE contactdate date;

declare temsno1,temsno2 smallint;

DECLARE temtid1,sstation1,astation1,carrigeid1 int;

declare done1 boolean default 0;

declare cur1 cursor for select PCardID from DiagnoseRecord where DStatus=1 ORDER BY PCardID;

-- 不能重复声明变量，循环里不能。。

DECLARE CONTINUE HANDLER FOR NOT FOUND set done1 =1;

open cur1;

FETCH cur1 into patientid;

while not done1 do

select TID,SStationID,AStationID,CarrigeID into temtid1,sstation1,astation1,carrigeid1

from taketrainrecord

WHERE PCardID=patientid;

SELECT SDate into contactdate from train WHERE train.TID=temtid1;

drop table if exists tmp1;

CREATE TEMPORARY TABLE tmp1 AS (SELECT SNo,SID FROM trainpass WHERE trainpass.TID=temtid1);

SELECT SNo into temsno1 FROM tmp1 WHERE tmp1.SID=sstation1;

SELECT SNo into temsno2 FROM tmp1 WHERE tmp1.SID=astation1;

-- SELECT SID FROM tmp1 WHERE trainpass.SNo BETWEEN temsno1 and temsno2;

-- mysql 不支持 Select .. into newtable

-- (下面这步骤分了两个，性能30倍。。。。)

-- drop table if exists contacttmp;

-- CREATE TEMPORARY TABLE contacttmp AS

-- (

-- select PCardID

-- from taketrainrecord,trainpass as trp1,trainpass as trp2

-- where taketrainrecord.TID=temtid1

-- and taketrainrecord.PCardID!=patientid

-- and CarrigeID=carrigeid1

-- and trp1.TID=temtid1

-- and trp2.TID=temtid1

-- and trp1.SID=SStationID

-- and trp2.SID=AStationID

-- and (trp1.SNo BETWEEN temsno1 and temsno2 or trp2.SNo BETWEEN temsno1 and temsno2)

-- );

drop table if exists contacttmp0;

CREATE TEMPORARY TABLE contacttmp0 AS

(

select PCardID,SStationID,AStationID

from taketrainrecord

where taketrainrecord.TID=temtid1

and taketrainrecord.PCardID!=patientid

and CarrigeID=carrigeid1

);

drop table if exists contacttmp;

CREATE TEMPORARY TABLE contacttmp AS

(

select PCardID

from contacttmp0,trainpass as trp1,trainpass as trp2

where

trp1.TID=temtid1

and trp2.TID=temtid1

and trp1.SID=SStationID

and trp2.SID=AStationID

and (trp1.SNo BETWEEN temsno1 and temsno2 or trp2.SNo BETWEEN temsno1 and temsno2)

);

call innerfillcontact(contactdate,patientid);

FETCH cur1 into patientid;

END WHILE;

CLOSE cur1;

end$$;

delimiter;

delete from traincontactor;

call fillcontact();

-- 验证接触表

select \* from taketrainrecord WHERE PCardID='230231197505243594'

UNION

select \* from taketrainrecord WHERE PCardID='330803200111012166';

select \* from taketrainrecord WHERE PCardID='110112194507080930'

UNION

select \* from taketrainrecord WHERE PCardID='140311195809071233';

-- 2.2数据更新

2.21

select count(\*) from taketrainrecord;

增

INSERT into taketrainrecord VALUES(99999,130623195203211991,10418,1648,1648,1,2,'A',1);

SELECT \* FROM taketrainrecord where RID=99999;

改

UPDATE taketrainrecord set SeatNo='B' WHERE RID=99999;

SELECT \* FROM taketrainrecord where RID=99999;

删

DELETE from taketrainrecord where RID=99999;

SELECT \* FROM taketrainrecord where RID=99999;

2.22

-- create table wuhantake like data\_mgr;

-- insert into wuhantake select \*from data\_mgr;

drop table if exists wuhantake;

CREATE table wuhantake as(

SELECT \*

FROM taketrainrecord

WHERE taketrainrecord.SStationID IN( SELECT SID

FROM station

WHERE station.SName='武汉'

)

);

2.24

drop table if exists test2\_2\_4;

CREATE table test2\_2\_4(

col1 int,

col2 int

);

INSERT into test2\_2\_4 VALUES(2,3);

INSERT into test2\_2\_4 VALUES(2,4);

INSERT into test2\_2\_4 VALUES(5,7);

INSERT into test2\_2\_4 VALUES(5,7);

SELECT \* from test2\_2\_4;

DELETE FROM test2\_2\_4 WHERE col1=2;

UPDATE test2\_2\_4 SET col1=6 WHERE col2=7;

SELECT \* from test2\_2\_4;

2.25

drop view patientrecord;

create view patientrecord as(

SELECT taketrainrecord.PCardID,passenger.PName,passenger.Age,

train.TName,train.SDate,CarrigeID,taketrainrecord.SeatRow,SeatNo

FROM passenger,taketrainrecord,train

WHERE taketrainrecord.SStatus=1

and taketrainrecord.PCardID=passenger.PCardID

and taketrainrecord.TID=train.TID

ORDER BY taketrainrecord.PCardID ASC , train.SDate DESC

);

2.26

DROP TRIGGER IF EXISTS DiagnoseT;

CREATE TRIGGER DiagnoseT

AFTER INSERT ON diagnoserecord

FOR EACH ROW

BEGIN

IF new.DStatus = 1 -- 确诊

THEN

INSERT INTO TrainContactor(CDate,CCardID,DStatus,PCardID)

SELECT TRO.SDate,TRO.PCardID,2,new.PCardID -- 疑似

FROM (SELECT new.PCardID,taketrainrecord.TID ,taketrainrecord.CarrigeID,taketrainrecord.SeatRow,train.SDate -- 病人14天内乘车记录的临时表

FROM taketrainrecord,train -- 存储病人ID，车次号，车厢号，座位排号，发车日期

WHERE taketrainrecord.PCardID = new.PCardID -- 病人的记录

AND train.TID = taketrainrecord.TID

AND taketrainrecord.SStatus = 1 -已乘车

AND DATE\_FORMAT(new.FDay,'%Y-%m-%d') >= Train.SDate -- 14天之内

AND DATE\_FORMAT(new.FDay,'%Y-%m-%d') < Train.SDate + 14)AS TRP(PCardID,TID,CarrigeID, SeatRow,SDate),

(SELECT taketrainrecord.PCardID,taketrainrecord.TID ,taketrainrecord.CarrigeID,taketrainrecord.SeatRow,train.SDate -- 除病人外其他人14天内乘车记录的临时表

FROM taketrainrecord,train -- 存储病人ID，车次号，车厢号，座位排号，发车日期

WHERE taketrainrecord.PCardID!=new.PCardID -- 不是病人的记录

AND train.TID = taketrainrecord.TID

AND taketrainrecord.SStatus = 1 -- 已乘车

AND DATE\_FORMAT(new.FDay,'%Y-%m-%d') >= Train.SDate -- 14天之内

AND DATE\_FORMAT(new.FDay,'%Y-%m-%d') < Train.SDate + 14)

AS TRO(PCardID,TID,CarrigeID, SeatRow,SDate)

WHERE (

(TRO.SeatRow = TRP.SeatRow ) OR -- 同一行

(TRO.SeatRow - 1 = TRP.SeatRow)OR -- 前一行

(TRO.SeatRow + 1 = TRP.SeatRow )) -- 后一行

AND TRO.CarrigeID = TRP.CarrigeID -- 同一车厢

AND TRO.TID = TRP.TID -- 同一车次

AND TRO.SDate = TRP.SDate; -- 同一天

UPDATE TrainContactor

SET DStatus = 1

WHERE TrainContactor.CCardID = new.PCardID

OR traincontactor.CCardID IN(SELECT PCardID FROM diagnoserecord WHERE diagnoserecord.DStatus = 1) ; -- 注意，为避免重复加入的情况，会检查诊断表是否已确诊

-- 更新密切接触者中感染者信息

END IF;

END;

2.26

-- 存储函数或触发器中不允许显式或隐式提交mysql隐式提交的sql语句“drop...”，"truncate table ..."

-- \*\*\*mysql 不能同时update or insert\*\*\*

delimiter //

drop trigger if exists contactriger;

CREATE TRIGGER contactriger AFTER INSERT ON DiagnoseRecord FOR EACH ROW

BEGIN

if DiagnoseRecord.DStatus=1

THEN CALL innner\_contactriger(new.PCardID,new.FDay);

end if;

END;//

delimiter;

delimiter //

drop trigger if exists contactriger2;

CREATE TRIGGER contactriger2 AFTER UPDATE ON DiagnoseRecord FOR EACH ROW

BEGIN

if new.DStatus=1

THEN CALL innner\_contactriger(new.PCardID,new.FDay);

end if;

END;//

delimiter;

触发器测试

诊断表有一条不得病数据

4 110101198304063574 2020-02-21 3

让他得病

UPDATE DiagnoseRecord set DStatus=1,FDay='2020-02-16' WHERE PCardID='110101198304063574';

查他乘车记录

SELECT \* FROM taketrainrecord WHERE PCardID=110101198304063574;

4 110101198304063574 34 1620 1621 1 4 E 0

查同车次 同车厢记录

SELECT \* FROM taketrainrecord WHERE TID=34 and CarrigeID=1;

没有其他记录..数据少了

SELECT count(PCardID),TID ,CarrigeID FROM taketrainrecord GROUP BY TID ,CarrigeID LIMIT 1000;

找到一个同车次同车厢 两个人的，， 39238车7车厢（乘车记录25k小了，车厢太空，随便搞不一定有）

SELECT \* FROM taketrainrecord WHERE TID=39238 and CarrigeID=7;

21 11010519470926382X 39238 1615 1572 7 10 F 0

3841 210421193405023013 39238 1571 1573 7 7 A 1

查这俩人诊断表

SELECT \* FROM DiagnoseRecord WHERE PCardID='11010519470926382X' or PCardID='210421193405023013';

21 11010519470926382X 2020-01-29 2

3841 210421193405023013 2020-02-29 3

都没病，让第一个得病 发病时间乘车时间后一点儿

SELECT train.SDate from train WHERE TID=39238;

2019-12-18

UPDATE diagnoserecord set DStatus=1,FDay='2019-12-24' WHERE PCardID='11010519470926382X';

SELECT count(\*) FROM diagnoserecord ;

查第是否进入接触表

SELECT \* FROM traincontactor WHERE traincontactor.CDate='2019-12-24';

其中有

2019-12-24 210421193405023013 2 11010519470926382X

表示触发器正确

SELECT FROM train WHERE TID=39238

SELECT \* FROM DiagnoseRecord LIMIT 10;

SELECT \* FROM taketrainrecord WHERE PCardID='110101194701201338';

delimiter $$

drop procedure if exists innner\_contactriger;

create procedure innner\_contactriger(in patientid char(18),in Fday0 date)

begin

declare contactorid char(18);

DECLARE contactdate date;

declare temsno1,temsno2 smallint;

DECLARE temtid1,sstation1,astation1,carrigeid1 int;

declare done1 boolean default 0;

declare cur1 cursor for

SELECT taketrainrecord.TID,taketrainrecord.CarrigeID,taketrainrecord.SStationID,taketrainrecord.AStationID

from taketrainrecord,train

WHERE taketrainrecord.PCardID=patientid

and taketrainrecord.TID=train.TID

and (train.SDate BETWEEN DATE\_SUB(FDay0,INTERVAL 14 day) and FDay0 );

DECLARE CONTINUE HANDLER FOR NOT FOUND set done1 =1;

open cur1;

FETCH cur1 into temtid1,carrigeid1,sstation1,astation1;

while not done1 do

SELECT SDate into contactdate from train WHERE train.TID=temtid1;

drop table if exists tmp1;

CREATE TEMPORARY TABLE tmp1 AS (SELECT SNo,SID FROM trainpass WHERE trainpass.TID=temtid1);

SELECT SNo into temsno1 FROM tmp1 WHERE tmp1.SID=sstation1;

SELECT SNo into temsno2 FROM tmp1 WHERE tmp1.SID=astation1;

drop table if exists contacttemp0;

CREATE TEMPORARY TABLE contacttemp0 AS

(

select PCardID,SStationID,AStationID

from taketrainrecord

where taketrainrecord.TID=temtid1

and taketrainrecord.PCardID!=patientid

and CarrigeID=carrigeid1

);

drop table if exists contacttemp;

CREATE TEMPORARY TABLE contacttemp AS

(

select PCardID

from contacttmp0,trainpass as trp1,trainpass as trp2

where

trp1.TID=temtid1

and trp2.TID=temtid1

and trp1.SID=SStationID

and trp2.SID=AStationID

and (trp1.SNo BETWEEN temsno1 and temsno2 or trp2.SNo BETWEEN temsno1 and temsno2)

);

call innerfillcontact2(contactdate,patientid);

FETCH cur1 into temtid1,carrigeid1,sstation1,astation1;

END WHILE;

CLOSE cur1;

end$$;

delimiter;

delimiter $$

drop procedure if exists innerfillcontact2;

create procedure innerfillcontact2(in contactdate date,in patientid char(18))

begin

##利用concat拼接sql,execute中执行拼接的sql

declare done2 boolean default 0;

declare contactorid char(18);

-- declare cur2 cursor for select PCardID from contacttmp;

-- set @sqlStr=concat('declare cur2 cursor for select PCardID from',tbNm);

-- PREPARE stmt from @sqlStr;

-- EXECUTE stmt;

declare cur2 cursor for select PCardID from contacttemp;

DECLARE CONTINUE HANDLER FOR NOT FOUND set done2=1;

open cur2;

FETCH cur2 into contactorid;

while not done2 do

INSERT INTO traincontactor VALUES(contactdate,contactorid,1+FLOOR (RAND() \* 3),patientid);

FETCH cur2 into contactorid;

end while;

CLOSE cur2;

end$$;

delimiter;

2.3

1）查询确诊者“张三”的在发病前14天内的乘车记录；

找一个发病的 李玉兰

SELECT \* FROM passenger,diagnoserecord WHERE diagnoserecord.DStatus=1 and passenger.PCardID=diagnoserecord.PCardID LIMIT 10;

李玉兰周志颖王小范胡冬松付强大郑文坚刘丹星莫梦莹郭宇刘珩

110101193412052028 李玉兰 0 86 1 110101193412052028 2019-12-14 1 2019-12-04

SELECT \* FROM taketrainrecord

WHERE PCardID in(SELECT PCardID FROM passenger WHERE passenger.PName='刘丹星')

AND TID in(SELECT train.TID FROM passenger,diagnoserecord,train

WHERE passenger.PName='刘丹星'

and diagnoserecord.PCardID=passenger.PCardID

and train.SDate BETWEEN DATE\_SUB(diagnoserecord.FDay,INTERVAL 14 day) and diagnoserecord.FDay );

-- 用户变量查要快一倍

SELECT \* FROM taketrainrecord

WHERE PCardID in(SELECT @pcardid1:=PCardID FROM passenger WHERE passenger.PName='刘丹星')

AND TID in(SELECT train.TID

FROM train,(SELECT @tmpday1:=FDay FROM diagnoserecord WHERE diagnoserecord.PCardID=@pcardid1) as zzz1

WHERE train.SDate BETWEEN DATE\_SUB(@tmpday1 ,INTERVAL 100 day)and '@tmpday1');

2）查询所有从城市“武汉”出发的乘客乘列车所到达的城市名；

SELECT @tmpwuhansid:=station.SID FROM station WHERE station.SName='武汉';

SELECT station.CityName FROM station

WHERE station.SID in(SELECT taketrainrecord.AStationID FROM taketrainrecord WHERE taketrainrecord.SStationID=@tmpwuhansid );

3）计算每位新冠患者从发病到确诊的时间间隔（天数）及患者身份信息，并将结果按照发病时间天数的降序排列；

select DiagnoseRecord.PCardID,PName,Sex,Age,DATEDIFF(DDay,FDay) as confirmday

from DiagnoseRecord,Passenger

where DiagnoseRecord.PCardID=Passenger.PCardID and DStatus=1

ORDER BY confirmday desc;

4）查询“2020-01-22”从“武汉”发出的所有列车；

select TName

from Station,Train

where Station.SID=Train.SStationID and Station.CityName='武汉' and Train.SDate='2020-01-22';

5）查询“2020-01-22”途经“武汉”的所有列车；

select distinct TName

from Train,TrainPass,Station

where Train.TID=TrainPass.TID and TrainPass.SID=Station.SID and Station.CityName='武汉'

and TrainPass.STime is not null and TrainPass.ATime is not null

and (DATE\_FORMAT(TrainPass.ATime,'%Y-%m-%d')='2020-01-22' or DATE\_FORMAT(TrainPass.STime,'%Y-%m-%d')='2020-01-22');

6）查询“2020-01-22”从武汉离开的所有乘客的身份证号、所到达的城市、到达日期；

-- a1 b1 武汉那天出发 a2 b2选出来到达的城市日期

SELECT PCardID,b2.CityName,DATE\_FORMAT(a2.STime,'%Y-%m-%d')

FROM taketrainrecord,trainpass as a1,trainpass as a2,station as b1,station as b2

WHERE taketrainrecord.TID=a1.TID and taketrainrecord.SStationID=a1.SID

and DATE\_FORMAT(a1.ATime,'%Y-%m-%d')='2020-01-22' and a1.SID=b1.SID and b1.SName='武汉'

and taketrainrecord.TID=a2.TID and taketrainrecord.AStationID=a2.SID and a2.SID=b2.SID ;

7）统计“2020-01-22” 从武汉离开的所有乘客所到达的城市及达到各个城市的武汉人员数。

-- 同6一致 加入selct count

SELECT b2.CityName,count(DISTINCT PCardID)

FROM taketrainrecord,trainpass as a1,trainpass as a2,station as b1,station as b2

WHERE taketrainrecord.TID=a1.TID and taketrainrecord.SStationID=a1.SID

and DATE\_FORMAT(a1.ATime,'%Y-%m-%d')='2020-01-22' and a1.SID=b1.SID and b1.SName='武汉'

and taketrainrecord.TID=a2.TID and taketrainrecord.AStationID=a2.SID and a2.SID=b2.SID

GROUP BY b2.CityName;

8）查询2020年1月到达武汉的所有人员；

-- 主要 武汉乘车占 时间在1月 between and

select Passenger.PCardID,PName,Sex,Age

from Passenger,TakeTrainRecord,TrainPass,Station

where Passenger.PCardID=TakeTrainRecord.PCardID

and TakeTrainRecord.AStationID=Station.SID

and TakeTrainRecord.TID=TrainPass.TID

and TakeTrainRecord.AStationID=TrainPass.SID

and Station.CityName='武汉'

and DATE\_FORMAT(TrainPass.STime,'%Y-%m-%d') between '2020-01-01' and '2020-01-31';

9） 查询2020年1月乘车途径武汉的外地人员（身份证非“420”开头）；

-- 3选武汉选时间 1 上车站信息 2 下车站信息 taketrainrecord贯穿 加distinct 否则经过几站有几次

-- 正则匹配 left(passenger.PCardID, 3)<>'420'

SELECT DISTINCT Passenger.PCardID,PName,Sex,Age,taketrainrecord.RID

FROM passenger,taketrainrecord,trainpass as tr1,trainpass as tr2,trainpass as tr3,station

where taketrainrecord.TID=tr1.TID and taketrainrecord.TID=tr2.TID and taketrainrecord.TID=tr3.TID

and tr3.SID=station.SID and station.CityName='武汉'

and DATE\_FORMAT(tr3.Atime,'%Y-%m-%d') BETWEEN '2020-01-01' and '2020-01-31'

and taketrainrecord.SStationID=tr1.SID and taketrainrecord.AStationID=tr2.SID

and tr3.SNo BETWEEN tr1.SNo AND tr2.SNo

and left(passenger.PCardID, 3)<>'420' and taketrainrecord.PCardID=passenger.PCardID;

前两条

110101200504220677 王连重 1 15 9rid

110114195202010533 邓汉源 1 68 74rid

SELECT @tmpwuhansid:=station.SID FROM station WHERE station.SName='武汉';

1597

SELECT \* FROM taketrainrecord WHERE rid=9;

9 110101200504220677 6249 1542 1535 4 4 C 0

SELECT \* FROM trainpass WHERE tid=6249

过武汉，时间对

SELECT \* FROM train WHERE tid=6249

6249 2020-01-31 D3251 3997 2127 2020-01-31 09:26:00 2020-01-31 19:37:00

10）统计“2020-01-22”乘坐过‘G007’号列车的新冠患者在火车上的密切接触乘客人数（每位新冠患者的同车厢人员都算同车密切接触）。

-- 接触表接触日期 2020-01-22 ，确诊者身份id-乘车记录-列车 限制tid而限制G007

-- 病人记录视图没下车占。。。 不好找，即使不限制g007也只有几个

SELECT traincontactor.PCardID,count(take2.PCardID)

FROM taketrainrecord as take1,taketrainrecord as take2,traincontactor,train

WHERE traincontactor.CDate='2020-01-22'

and traincontactor.PCardID=take1.PCardID and take1.PCardID<>take2.PCardID

and take1.TID=train.TID and train.TName='G007'

and take1.TID=take2.TID and take1.CarrigeID=take2.CarrigeID

GROUP BY take1.PCardID

11）查询一趟列车的一节车厢中有3人及以上乘客被确认患上新冠的列车名、出发日期，车厢号；

SELECT TName,Train.SDate,CarrigeID

FROM Train,TakeTrainRecord,DiagnoseRecord

WHERE DiagnoseRecord.DStatus = 1 -- 已确诊

AND TakeTrainRecord.PCardID = DiagnoseRecord.PCardID -- 乘客已确诊

AND TakeTrainRecord.TID = Train.TID -- 车次

AND TakeTrainRecord.SStatus = 1 -- 已乘坐

GROUP BY CarrigeID,TName,Train.SDate

HAVING COUNT(\*)>=3;

12）查询没有感染任何周边乘客的新冠乘客的身份证号、姓名、乘车日期；

-- 需要考虑这个乘客的乘车区间，在这个乘车区间内所有周边乘客是否感染？

SELECT DISTINCT passenger.PCardID,passenger.PName,train.SDate

FROM train,TakeTrainRecord,DiagnoseRecord,traincontactor,passenger

WHERE train.TID = TakeTrainRecord.TID -- 车号统一

AND TakeTrainRecord.PCardID = DiagnoseRecord.PCardID -- 被确诊的乘客在乘车记录中

AND taketrainrecord.SStatus = 1 -- 已上车

AND passenger.PCardID = TakeTrainRecord.PCardID -- 被确诊的乘客在乘客信息中

AND DiagnoseRecord.DStatus = 1 ;-- 确诊

AND diagnoserecord.DID NOT IN(SELECT traincontactor.PCardID -- 筛出周边旅客被感染的

FROM traincontactor

WHERE traincontactor.DStatus = 1 );

13）查询到达 “北京”、或“上海”，或“广州”（即终点站）的列车名，

要求where子句中除了连接条件只能有一个条件表达式； 利用 in

SELECT DISTINCT train.TName

FROM station,train

WHERE station.SID=train.AStationID

and station.CityName IN('广州','上海','北京');

14）查询“2020-01-22”从“武汉站”出发，然后当天换乘另一趟车的乘客身份证号和首乘车次号，

结果按照首乘车次号降序排列，同车次则按照乘客身份证号升序排列；

-- 换乘车 当天第二个车上车站和第一趟车下车占相同 根据两个tid sid 限制时间同一天 晚于第一个

-- SELECT @tmpwuhansid:=station.SID FROM station WHERE station.SName='武汉';

-- 多一句吧，简单 或者直接用武汉的1597

SELECT a1.TID

FROM taketrainrecord as tak1,taketrainrecord as tak2,trainpass as a1,trainpass as a2,station as b1,station as b2

WHERE tak1.TID=a1.TID and tak1.SStationID=1957 and tak1.AStationID=a1.SID

and tak1.AStationID=tak2.SStationID and tak2.TID=a2.TID and tak2.SStationID=a2.sid

and DATE\_FORMAT(a1.STime,'%Y-%m-%d')=DATE\_FORMAT(a2.ATime,'%Y-%m-%d')

and a1.STime<a2.ATime

ORDER BY a1.TID

;

15）查询所有新冠患者的身份证号，姓名及其2020年以来所乘坐过的列车名、发车日期，要求即使该患者未乘坐过任何列车也要列出来；

SELECT diagnoserecord.PCardID,passenger.PName,train.TName,train.SDate

FROM diagnoserecord,train,taketrainrecord,passenger

WHERE diagnoserecord.DStatus=1

and diagnoserecord.PCardID=taketrainrecord.PCardID

and taketrainrecord.TID=train.TID

and train.SDate>'2019-12-30'

and diagnoserecord.PCardID=passenger.PCardID;

16）查询所有发病日期相同而且确诊日期相同的病患统计信息，包括：发病日期、确诊日期和患者人数，结果按照发病日期降序排列的前提下再按照确诊日期降序排列。

SELECT diagnoserecord.FDay,diagnoserecord.DDay,count(diagnoserecord.PCardID)

FROM diagnoserecord

GROUP BY diagnoserecord.DDay,diagnoserecord.FDay

ORDER BY diagnoserecord.FDay DESC ,diagnoserecord.DDay DESC

CREATE table zz1(fday date);

INSERT into zz1 VALUES ('2020.1.29');

SELECT DATE\_SUB(zz2.fday,INTERVAL 100 day) aa1

SELECT now() aa2;

ALTER TABLE zz1 RENAME to zz2;

SET FOREIGN\_KEY\_CHECKS = 1;

## 存储管理sql

set time\_zone = '+8:00';

set global time\_zone='+08:00';

set @@global.auto\_increment\_increment = 1;

set @@auto\_increment\_increment =1;

1、职员表【姓名，工号，职位】

工号为主码

职务包括 审核人员 销售人员 采购人员

CREATE DATABASE if NOT EXISTS StorageManagement;

USE StorageManagement;

CREATE TABLE staff(

sname char(10),

sno char(6) PRIMARY KEY,

position char(4)

)

INSERT into staff VALUES

('李采购','c00001','采购'),

('王采购','c00002','采购'),

('章审计','s00001','审计'),

('孙销售','x00001','销售'),

('林销售','x00002','销售');

2、账户表【账号（同工号），密码】

账号为主码，参考外键职员表工号

CREATE TABLE account(

sno char(4) PRIMARY KEY,

spswd char(6),

FOREIGN KEY (sno) REFERENCES staff(sno)

);

3 库存表【货号，货名，存放地，库存数量，生产厂家】货号为主码

DROP TABLE IF EXISTS inventory;

CREATE TABLE inventory(

ino int PRIMARY KEY,

iname CHAR(10) ,

ilocation char(10),

inum int,

iproducer char(10),

CHECK(inum>=0)

);

INSERT into inventory VALUES

(1,'可口可乐','武汉1仓',100,'某饮料厂'),

(2,'怡宝','武汉1仓',200,'某饮料厂'),

(3,'农夫山泉','武汉1仓',300,'某饮料厂'),

(4,'脉动','武汉1仓',200,'某饮料厂'),

(5,'冰红茶','武汉1仓',100,'某饮料厂'),

(6,'绿茶','武汉1仓',100,'某饮料厂');

3、进货单【单号，货号，货名，下单时间，结单时间，完成状态，采购人工号，审核人工号】

单号为主码

货号外键参考库存表货号，采购人，审核人外键参考职员表。

完成状态 0，1，0代表待审核，1代表完成，完成的订单不可修改

-- mysql> set @@global.auto\_increment\_increment = 1;

-- mysql> set @@auto\_increment\_increment =1;

--

drop TABLE IF EXISTS buy;

CREATE TABLE buy (

bno int PRIMARY KEY auto\_increment, -- 主键自增 序号

bgno int ,

bgname char(10) ,

bnum int,

bstatus smallint, -- 完成状态 0，1，0代表待审核，1代表完成，

bstime datetime,

betime datetime,

bbuyer char(6) ,

bchecker char(6),

FOREIGN KEY (bgno) REFERENCES inventory(ino),

FOREIGN KEY (bbuyer) REFERENCES staff(sno),

FOREIGN KEY (bchecker) REFERENCES staff(sno),

CHECK(bnum>0)

)

4、

单号为主码

货号外键参考库存表货号，销售人，审核人外键参考职员表。

完成状态 0或1或2，0代表待审核，1代表完成，2代表审核完成但是缺货。 完成的订单不可修改，当补货后达到所需出货数量时自动出货，完成状态变为1。

出货单【单号，货号，货名，下单时间，结单时间，完成状态，销售工号，审核人工号】

drop TABLE IF EXISTS sell;

CREATE TABLE sell(

sno int PRIMARY KEY auto\_increment, -- 主键自增 序号

sgno int ,

sgname char(10) ,

snum int,

sstatus smallint, -- 完成状态 0，1，2

-- 0代表待审核，1代表完成，2代表审核完成但是缺货，只有0状态可被审核

sstime datetime,

setime datetime,

ssellor char(6) ,

schecker char(6),

FOREIGN KEY (sgno) REFERENCES inventory(ino), -- 外键参照库存商品号

FOREIGN KEY (ssellor) REFERENCES staff(sno), -- 外键参照职员工号

FOREIGN KEY (schecker) REFERENCES staff(sno),

CHECK(snum>0) -- 自定义约束

)

5缺货表【缺货货号，缺货数量】

货号为主键，参考库存表货号。

drop table if EXISTS lack;

CREATE TABLE lack(

lno int,

lname char(10),

lnum int,

PRIMARY KEY (lno),

FOREIGN KEY(lno) REFERENCES inventory(ino)

)

-- mysql不支持语句级别触发器

-- 采购通过审核后。判断是否缺货，处理库存和缺货

DROP TRIGGER IF EXISTS TrigerBuy;

CREATE TRIGGER TrigerBuy

AFTER UPDATE ON buy

FOR EACH ROW

BEGIN

IF old.bstatus=0 AND new.bstatus=1 -- 审核通过

THEN

set @lacknum=(SELECT IFNULL(lnum,0) FROM lack WHERE lno=new.bgno); -- 查缺货数量，不缺为0

if @lacknum=0

then UPDATE inventory set inum=inum+new.bnum WHERE ino=new.bgno; -- 不缺货 增库存

else set @changenum=new.bnum-@lacknum;

if @changenum>=0 -- 缺货 新货凑够发货 更新库存，删缺货表记录，缺货单状态改为1完成

then

UPDATE inventory set inum=@changenum WHERE ino=new.bgno;

UPDATE sell set sstatus=1 WHERE sgno=new.bgno and sstatus=2;

DELETE FROM lack WHERE lno=new.bgno;

else UPDATE lack set lnum=-@changenum WHERE lno=new.bgno; -- 缺货还不够发货，增库存 减缺货量

UPDATE inventory set inum=inum+new.bnum WHERE ino=new.bgno;

end if;

end if;

END IF;

END;

-- INSERT INTO buy VALUES(7,2,'怡宝',30,0,now(),now(),'x00001','s00001');

-- UPDATE buy set bstatus=1 WHERE bno=7;

-- INSERT INTO buy VALUES(8,1,'可口可乐',40,0,now(),now(),'x00001','s00001');

-- UPDATE buy set bstatus=1 WHERE bno=8;

-- INSERT INTO buy VALUES(6,1,'可口可乐',500,0,now(),now(),'x00001','s00001');

-- UPDATE buy set bstatus=1 WHERE bno=6;

--

-- UPDATE sell set sstatus=2 WHERE sgno=1 and sstatus=1;

-- DELETE FROM lack WHERE lno=1;

-- 销售触发 审核后销售订单状态变1 ，如果货足，发货库存减，否则订单状态改为2，填缺货表

DROP TRIGGER IF EXISTS trigerSell;

CREATE TRIGGER TrigerSell

BEFORE UPDATE ON sell

for EACH ROW

BEGIN

IF old.sstatus=0 AND new.sstatus=1 -- 由缺货变发货 由另一个触发器处理

then

SET @tempinum=(SELECT inum FROM inventory WHERE ino=new.sgno);

IF new.snum<=@tempinum

then UPDATE inventory set inum=inum-new.snum WHERE ino=new.sgno;

ELSEIF new.snum>@tempinum

then set new.sstatus=2; -- 取货更新状态1变为2，修改这个必修before update

INSERT INTO lack VALUES (new.sgno,new.sgname,new.snum-@tempinum);

end if;

end if;

end;

-- INSERT INTO sell VALUES(0,2,'怡宝',20,0,now(),now(),'x00001','s00001');

-- INSERT INTO sell VALUES(0,1,'可口可乐',300,0,now(),now(),'x00001','s00001');

-- UPDATE sell set sstatus=1 WHERE sno=4;

-- UPDATE sell set sstatus=1 WHERE sno=5;

--